Volume 9 Issue 1 May 1994

ICL

Ingenuity

THE TECHNICAL JOURNAL

Ingenuity™

The Technical Journal published twice a year by International Computers Limited.

1994 subscription rates:

	UK and Europe	Rest of World
Annual subscription	£60	$120
Single issues	£36	$72

ISSN 1354-9952

Ingenuity

Volume 9 Issue 1

Contents

Editorial Note

This issue is largely devoted to matters connected with client-server and distributed computing - subjects of great interest at this time. Their importance is underlined in the Foreword to this issue by Ninian Eadie, ICL Group Executive Director, Technology.

There was an unexpectedly heavy response to a call for papers on the subject and, as a result, several papers on more specialised aspects such as security have had to be held over.

This issue marks an important change in publications arrangements. The Journal is now published by ICL itself and has been renamed **Ingenuity**: the cover of the Journal has been altered, to make it more interesting: less obvious changes have been made to the typography so that it is easier to read: the basic A5 format popular with readers has, however, been retained. Regrettably, it has been necessary to make a (discounted) charge now payable by readers who previously received a complimentary copy.

There will be no change in the editorial policy of printing papers designed to appeal to technically minded readers who may not enjoy deep knowledge of the area covered.

Foreword

This issue of **Ingenuity**, the ICL Technical Journal, is majoring on Client/Server (C/S) systems.

It is less than a year since we set up a separate Client-Server Systems organisation and in that time C/S has become one of the most pervasive terms in our industry. Yet neither the term not the concept is new. As soon as the dumb terminal gained intelligence, programmers began to move logic downstream to improve response times. This was quickly elaborated into systems with local area networks and shared services, like DRS 20 (1981) which can claim to have been among the first C/S products.

However, at this stage, the IBM personal computer had not even been invented, and the explosive growth in personal computing which followed had to take place before the industry could fully exploit these architectural concepts.

While the growth in personal computers was fuelled by rapid advances in microprocessor technology, and the dramatic improvements in cost performance which this made possible, it is important to recognise that there were also political reasons for its enthusiastic adoption. These changes were not unique to our industry, the eighties saw a paradigm shift from the centrally planned to the deregulated, from communism to democracy. IT users, alienated by centralised computing, saw the move to end user computing in the same terms and embraced the new world regardless of its obvious shortcomings.

Those shortcomings became increasingly evident. MIPS are cheap, but the hidden burden of end user support is expensive. Mix and match is fine in theory, but does not always work well in practice. There are very real problems about data back-up, virus protection, security and data integrity. Besides, not all tasks are personal, many require teamwork.

The over-blown expectations of personal computing, and the subsequent disenchantment, have been important factors in leading to the rapid take-up of C/S. On the one hand C/S allows the end user to keep the sub-second response times and graphical user interface to which he has become addicted, while on the other it offers the enterprise the prospect of a more disciplined approach to systems management and data sharing.

In the circumstances it was unfortunate, but perhaps inevitable, that the Press wrote this up as a battle between mainframes and PCs. In reality, many servers will require those industrial strength characteristics which had been evolved in mainframes, at much pain and expense.

Furthermore, the hard earned skills developed to manage large mainframe projects are precisely those that are now required to integrate C/S systems, many of which are every bit as complicated. Certainly, there are new skills to learn, but it would be a great pity if we were to throw the baby out with the bath water.

Finally, C/S is no passing fad. It is the architecture of the long term future. As competition between the PTTs and the entertainment industry drives down costs, the network will reach out beyond the enterprise, and we will look further and further afield for services. Already programs are written in Bangalore, and networks managed from the United States. The Superhighway will connect the computer on our desk to remote servers which will supply us with full motion video, desktop conferencing, and geographic information, as well as many conventional application services. Client/Server has migrated from the department to the enterprise, and is about to go global.

Ninian Eadie

ICL Group Executive Director, Technology

Client-server architecture

John Brenner
OPEN*framework* Division, ICL, Bracknell, UK

Abstract

Client-server is an essential part of the vision for open systems integration that is expressed in OPEN*framework* and ICL products. This paper explains what client-server is and why it is important, and what client-server technology is and how it is used. This is the background within which the final part of the paper describes client-server architecture.

1 Introduction

Client-server is very fashionable. As such, it might be just a temporary fad; but there is general recognition that it is something fundamental and far-reaching; for example, the Gartner Group, who are leading industry analysts in this field, have predicted that

"*By 1995 client-server will be a synonym for computing.*"

We therefore need a clear understanding of what it is and why it is important.

1.1 Business meaning of client-server

Client-server is generally perceived to be the next step forward in the operational effectiveness of business information systems. This is illustrated in figure 1, which indicates cumulative gains from a succession of innovations.

Business computing started in the 1960s with batch processing. The main innovation in the 1970s was on-line transaction processing (OLTP), which brought information technology (IT) to the desktop, and made it an integral part of business processes. Batch processing and OLTP in combination continue to be at the core of most enterprise's information systems. Then in the 1980s came personal computing, which made IT universally affordable and dispersed it throughout business enterprises. Now in the 1990s, client-server is generally perceived to be the way of integrating the separate parts of information systems back together. That is its role and its importance.

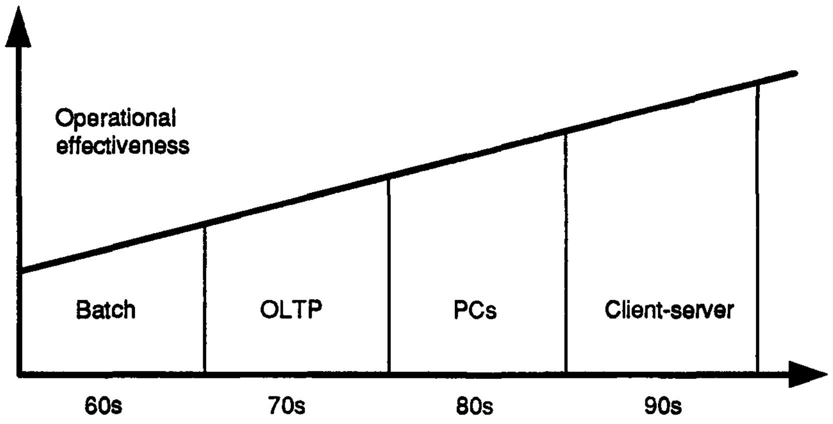


Figure 1 Perceived business impact of client-server

In these circumstances client-server (or client/server) has become a popular brand name that is applied to almost every kind of product, and to all manner of business and technical insights and marketing messages. This tends to drain it of specific meaning; but in doing so, actually confirms its near-universal applicability.

1.2 Technical meaning of client-server

A useful starting point for understanding client-server is the informal definition used by the Gartner Group:

> "*Client-server is the splitting of an application into tasks that are performed on separate computers, one of which is a programmable workstation (e.g. a PC).*"

This definition says that client-server is about distributed computing and software architecture (applications are split into tasks that may be on separate computers). It echoes the vital point that client-server is <u>the</u> way to integrate PCs into all kinds of information systems.

2 Client-server Technology

Client-server technology is best understood if we discuss it in five areas:

1. personal platforms
2. server platforms
3. client-server middleware
4. client-server applications
5. client-server tools and services

Each of these areas is distinctive, although there can be overlap between them.

The term *platform* is used here to refer to a computer platform that is a complete combination of hardware and operating system software.

2.1 Personal platforms

Personal platforms are perhaps the most distinctive area of client-server technology. We define a **personal platform** as:

> **A computer platform which is connected to a network, provides a consistent and intuitive user interface and assisting a personal user to accomplish tasks on behalf of the enterprise.**

These characteristics are illustrated in figure 2. Personal platforms are relatively inexpensive and immensely powerful, and there is a wide choice of suppliers. Many different kinds of computers can be personal platforms (e.g. MS/DOS PC, Windows PC, OS/2 PC, UNIX™ workstation, Apple Macintosh, and various hand-held devices); but the most common case today is an IBM-compatible PC with the Microsoft™ Windows operating system.

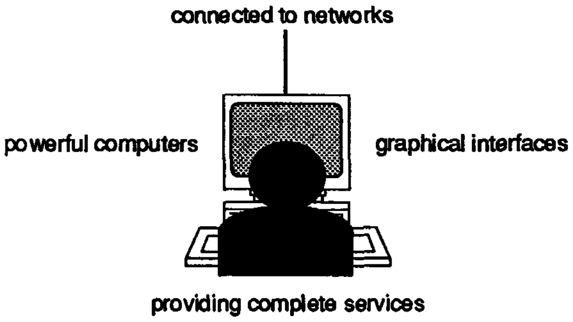


Figure 2 Characteristics of a client platform

Such platforms are now universally affordable wherever they are needed. This has turned the architecture of computer systems inside-out: the old focus was scarce resources in the central machine, remote from its users; the new focus is the abundant personal resources now at the fingertips of each individual user. This trend has ever-increasing force, because PC price/performance ratios continue to improve by a factor of two every eighteen months or so.

This change of focus aligns with changes in business structure: organisational hierarchies are being flattened, decision-making authority is being devolved, and processes that were formerly provided by office staff can now be provided by IT-enabled processes. A combined effect of these business and technical trends is *personal empowerment* of the individual at the desk.

PCs provide personal productivity and independence, but this individuality, multiplied by huge numbers of PCs, can also create anarchy.

Client-server helps to resolve these problems. The clients use shared resources (provided on server platforms), not just personal resources; client-server structure enables all the software and hardware resources to be under architectural and management control. It transforms personal computing into inter-personal computing and enterprise-wide computing. These characteristics help to create order, workgroup cohesion, productivity, and flexibility of business process.

Although personal platforms are the main economic and technical driving force for the move to client-server, they are only the first of the five technical ingredients identified at the start of section 2.

2.2 Server platforms

We define a **server platform** as:

> **A computer platform on which software provides IT services for use elsewhere in the system.**

Ultimately the services are for use at personal platforms; but services are also provided for use at other server platforms. A server platform may provide services via dependent terminals that do not qualify as personal platforms.

Almost all kinds of computer platform can act as server platforms. Therefore, there are many different suppliers, and many possible kinds of server platforms, from super computers to PCs. Each is good for particular kinds of workloads, for different qualitative requirements, and in different areas of the price and performance spectrum. User enterprises can select different platforms to match different needs.

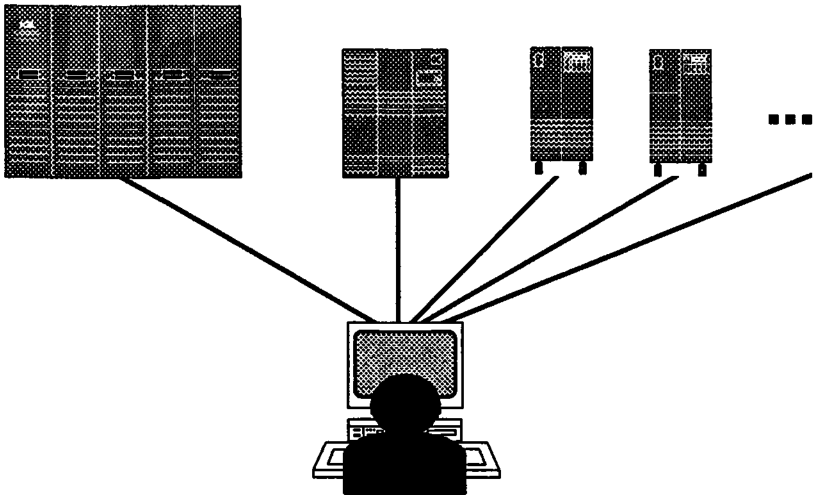


Figure 3 Many server platforms to choose from

This breadth of choice is illustrated in figure 3, which shows that the user at a personal platform may have access to services on many server platforms. This also illustrates the shift of focus onto the individual user at a personal platform, who may now choose IT services from many different sources elsewhere in the computer network.

The polarisation of systems into client and server platforms recognises distinctions between personal and shared resources. Each personal platform is an independent personal resource, which may be mobile and is exposed to risks of accidental loss or damage. Conversely, a server platform provides a protected, fixed, and carefully managed environment for shared resources.

Even where the same technology is used for both client and server platforms (e.g. PCs with the same kind of hardware and operating system), these distinctions between personal and shared resources should be made. In the limit, the same machine may be both a personal platform and a server platform (e.g. in a peer-to-peer network; see 3.2). As always, the server role brings obligations to guarantee availability and integrity of the shared resources.

2.3 Client-server middleware

We define **client-server middleware** as:

> **Packaged software to support the separate parts of client-server application software and enable them to work together.**

This is by far the most complex area of client-server technology. By concentrating the complexity here we are able to keep the other areas relatively simple. It includes many kinds of function, each of which may itself be distributed, and most of which are inter-related. Some of the main areas are:

- networking services
- distributed application services
- distributed systems management
- distributed security
- distributed object management
- user interface management
- print management
- data management
- transaction management
- workflow management

Figure 4 is a symbolic representation of this middleware support for client-server application software. It emphasises the importance of middleware in enabling client-server technology to operate across the whole business scope relevant to the user's tasks. This may involve interaction across departmental and functional boundaries, and perhaps across enterprise boundaries.

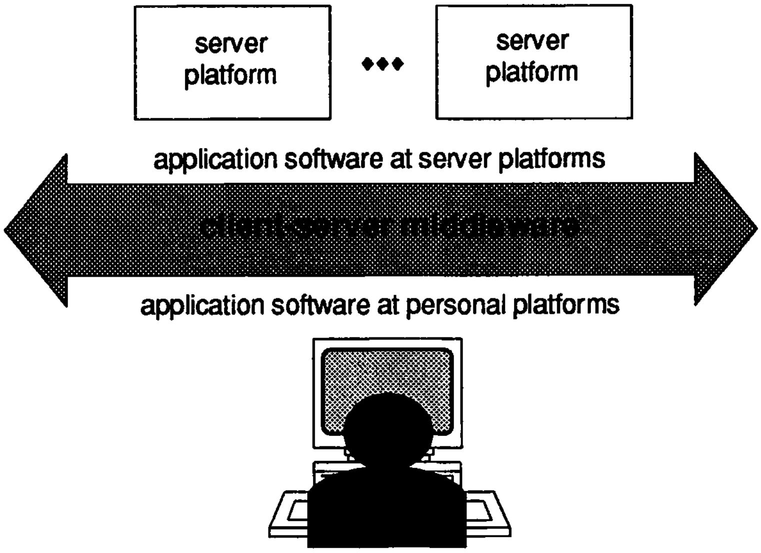

server platform
server platform
application software at server platforms
client-server middleware
application software at personal platforms

Figure 4 Client-server middleware

Some important areas of client-server middleware technology are described in [Archer, 1994] and [Duxbury, 1994].

2.4 Client-server applications

We define a client-server application as:

> **An application system in which logically separate software components are integrated together via client-server relationships.**

In a client-server relationship, one part of an application (the client end) uses a service provided by the other part (the server end). The latter is often a shared resource, used by many clients. Although integrated together via the client-server relationship, the parts remain separate. We refer to them as being logically separate because they need not be physically remote from one another (they might be in the same computer).

We describe client-server application software here in three steps: splitting an application (in 2.4.1), joining separate applications together (in 2.4.2), and distributed application structure (in 2.4.3).

2.4.1. Splitting an application

There are many ways of partitioning application software into separate components. However, the content of most applications can usually be classified under three different technical headings: data management, application logic and presentation. This is illustrated in figure 5.

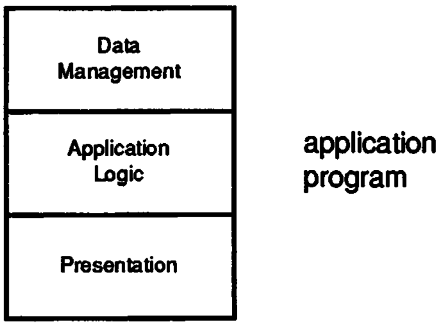


Figure 5 Application software modularity

If the application is to be split into two parts (one part on a client platform, the other on a server platform), the split can be made at either of the two boundaries between functions, or inside one of the three functions. Consequently there are five main ways of splitting a centralised or personal application into two parts between which there is a client-server relationship. This is the basis of the popular classification into five client-server styles, which is promoted by the Gartner Group. It is illustrated in figure 6.

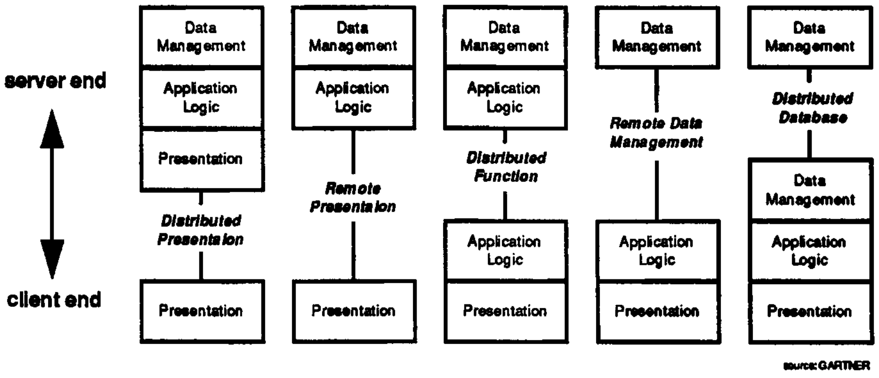


Figure 6 Five generic styles of basic client-server structure

The details need not concern us here. The important point is that different styles suit different needs and circumstances:

- The two styles on the left of the diagram are typical of *centralised interactive applications* that have been adapted to client-server by means of graphical interface technology, terminal emulation, etc.

- The style in the middle of the diagram is typical of *object-oriented distributed applications* and *distributed TP applications* in which data and function are encapsulated together behind application interfaces
- The two styles on the right of the diagram are typical of *data-centred applications* using client-server 4GL development tools and relational database products

Specific examples are described in [Day, 1994].

Some applications combine all three areas of function (presentation, application logic and data management) at the personal platform. Also, different styles may occur in combination at the same platform.

2.4.2 Joining applications together

One of the great strengths of client-server is the ability to join separate applications together. This can be done in many ways; but upon the principles used in 2.4.1, there are essentially three levels at which applications can interface with one another. This is illustrated in figure 7.

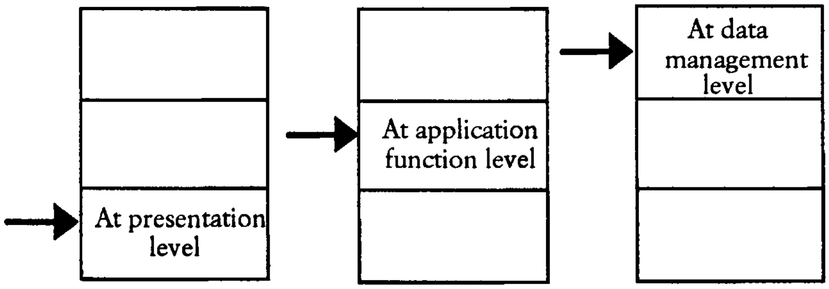


Figure 7 Three levels at which applications can be joined together

The main characteristics and advantages and disadvantages of these three approaches are:

- **At presentation level:** Interaction at this level is achieved via direct data exchange (DDE) within a window management system, or via *scripting*; see [Duxbury, 1994], in which software uses an application's user interface by simulating a human user. This kind of technique is often referred to as *screen scraping*. It is very useful for accessing legacy applications, but leads to software maintenance problems if the user interfaces need to change.
- **At application function level:** Interaction at this level is in terms of business functions. Therefore, the inter-application requests are about the business meanings of the application (and not its presentation or database encoding). This has the advantage of keeping their internal designs separate from their external interactions. There are fewer software maintenance problems.

- **At data management level:** Interaction at this level is by direct access to the other application's database. This is common practice, but leads to software maintenance problems when application data structures change.

The first and third approaches inhibit potential for change, the second does not. Further distinctions can be made between direct and indirect interaction between applications, synchronous and asynchronous interaction, and externally programmed interaction and internally programmed interaction.

2.4.3 Distributed application structure

Distributed applications are evolving towards richly-connected network structures of the kind illustrated in figure 8. The circles represent separate software components, and the lines represent client-server relationships between them. This is typical of the kind of structure that results from use of object-oriented design and distributed object management.

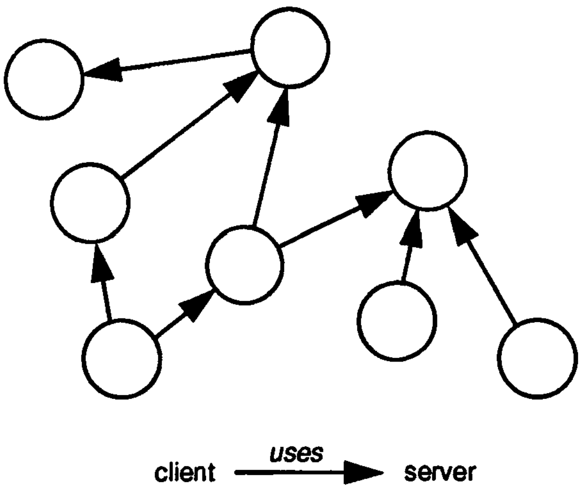


Figure 8 Complex distributed application

There is also large-scale structure of distributed application systems (within which the individual client-server relationships occur). Typically, three tiers of application software can be discerned in the large-scale structure:

- Front tier: Application software (and databases) at personal platforms, providing all kinds of application services, using local resources and remote resources. Typically, the platforms are PCs. This tier is where the greatest amount of computer power and of new application software is now being deployed.

- **Middle tier:** Application software (and databases) at server platforms, providing the back-end of personal applications, shared workgroup services and task-oriented services. Typically, the platforms are UNIX or PC. This tier provides rapid adaptation to business process change, without needing changes to the back tier. It puts boundaries around the turbulence and uncertainty generated in the volatile world at the first tier, where all the users are. It also provides lateral linkage across the enterprise (e.g. electronic mail services).
- **Back tier:** Application software and databases at server platforms providing corporate information services. These are usually functionally partitioned (e.g. accounts, manufacturing, personnel). Typically, the platforms are mainframes. This tier provides the core of shared and long-lived information assets that everything else depends on. There are strong guarantees of data integrity, and the applications and databases are stable, and their design changes rather slowly.

This structure separates different kinds of concerns which used to be bundled together in centralised computing.

2.5 Client-server tools and services

This is the fifth and final area of client-server technology.

Client-server systems may be complex, but with well-integrated systems and well-designed user interfaces the technical complexity should not be visible to the user; it is essentially a problem for the application developer and service provider. They need software development tools and professional services to help manage and hide this complexity. Many of the tools and services needed are the same as always, but there are also needs specific to client-server systems.

An important general point is that for packaged ("shrink-wrapped") application software, the user enterprise does not need program construction tools. Packaged client-server application products are now becoming widely available (e.g. distributed office and groupware applications, business accounting applications, personnel and payroll applications).

Another important trend is that different tools (and languages) are needed for different parts of modular application systems. The main distinctions are:

- **User interface:** languages and tools for construction of graphical user interfaces and any application logic intimately associated with them; e.g. GUI tools and Visual Basic.
- **Database:** languages and tools for the construction of databases, file systems and object stores, and construction of the application logic intimately associated with them; e.g. Data Manipulation Languages and Relational Database 4GLs.

- **Business logic:** languages and tools for the construction of application logic that is logically separate from user interfaces and databases; e.g. COBOL.
- **Distributed processing:** languages and tools specialised for distributed processing, and for spanning all the above functional areas (and other technological and organisational boundaries); e.g. Remote Procedure Call (RPC) tools.
- **System management:** methods and tools for electronic distribution of software, and operation and tuning of client-server systems.

Most of these tools are associated with the corresponding areas of middleware.

3 Client-server architecture

By looking back over the technology described in the previous section, three kinds of client-server architecture can be discerned.

3.1 Basic client-server

We use the term *basic* client-server for the kind of two-way splits considered in sections 1.2 and 2.4.1.

In basic client-server architecture, a personal or centralised application is split into two parts: a client part on a personal platform, and a server part on a server platform. The latter is often a shared resource, such as a filing service, a printing service, a database, or some application-specific function. The terms *client* and *server* are used to refer to the hardware platforms and the application software components (often somewhat ambiguously).

Basic client-server architecture is illustrated in figure 9 (and has already been shown in more detail in figure 6).

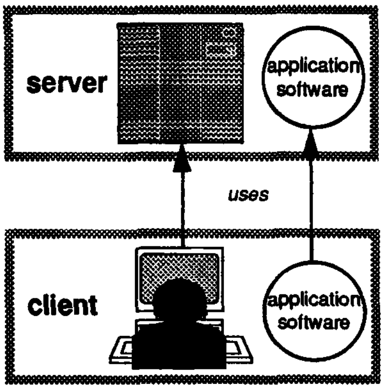


Figure 9 Basic client-server architecture

Basic client-server configurations are normally organised around a local area network (LAN). The whole assembly is usually described as a PC-LAN, and consists of many PCs for personal use (personal platforms), plus one or more shared PCs (server platforms). The local server platforms on these PC-LANs usually provide gateways into enterprise-wide and external networks, and to the servers on them. This is illustrated in figure 10.

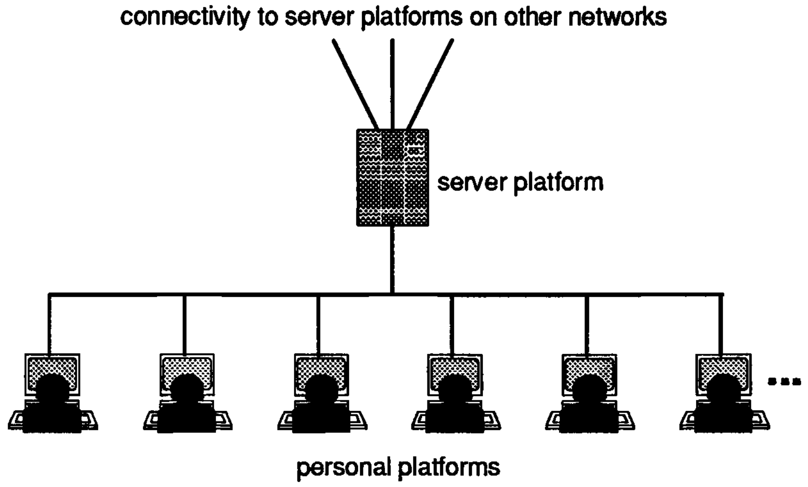


Figure 10 A typical PC-LAN

Although primarily expressed in terms of PCs and PC-LANs, these basic client-server concepts are applicable to all kinds of computers and networks (e.g. PCs, UNIX, mainframes, LANs and WANs).

3.2 Beyond the basics

Beyond basic client-server there is peer-to-peer processing, co-operative processing and standalone processing.

The term **peer-to-peer processing** is used to refer to configurations in which there are no server platforms, and the server parts of applications are located on personal platforms. Networks operating on this basis are referred to as **peer-to-peer networks.** This is a low-cost way of implementing small PC-LANs, etc.; but the lack of separate server platforms reduces system integrity and leads to system management difficulties.

The term **co-operative processing** is used to refer to configurations in which application software is distributed over separate server platforms, and the client and server ends of interactions are both on server platforms. This includes interaction between separate applications, not just between parts of the same application.

The term **stand-alone processing** is used to refer to configurations in which all parts of an application are on one platform (usually a personal platform). Any client-server relationships between the parts are not externally visible.

People also use the terms peer-to-peer and co-operative processing interchangeably, and with various other meanings. This causes confusion and misunderstandings. There are also various other less well known formulations such as server/requester and producer/consumer. All the main formulations are illustrated together in figure 11.

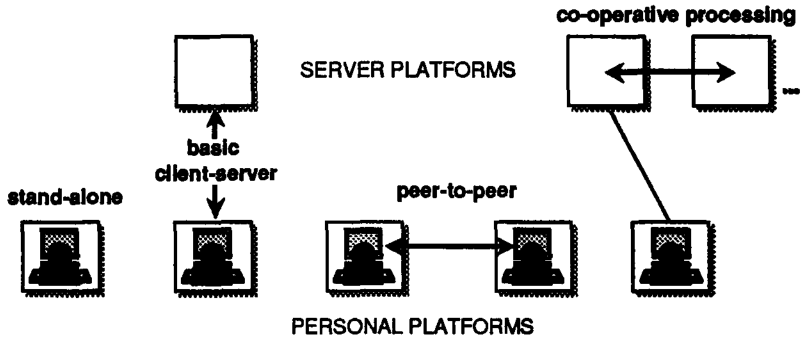


Figure 11 Various formulations of client-server system structure

Unfortunately, many people sharply differentiate the other concepts from client-server (by which they really mean *basic* client-server). This obscures the vital point that all are variants within one unified structure: client-server architecture. It also leads to misleading statements to the effect that client-server (meaning basic client-server) is defunct, and is being superseded by other techniques such as co-operative processing.

3.3 General client-server architecture

A fundamental limitation of basic client-server and of all the formulations in 3.1 and 3.2 is that they define software configuration in ways dependent on hardware configuration. Furthermore, it is often ambiguous whether the terms client and server refer to the software or the hardware.

To escape from these limitations and ambiguities, client-server relationship in software should be defined independently of software location, and independently of any classification of the underlying hardware as clients or servers.

The essential clarification is that client and server are *roles* in which services are used and provided (respectively), and these roles occur in a relationship between autonomous *building-blocks*. In such a relationship, one of the participants uses a service (it has the client role) and another provides the service (it has the server role). This is a client-server relationship. Large and flexible configurations can be built up by combination of these simple concepts. This is illustrated in figure 12.

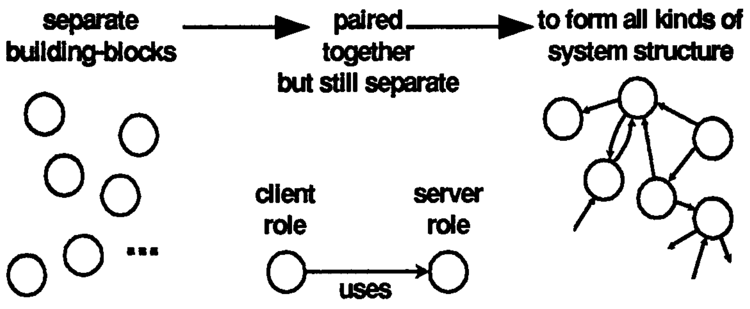

separate building-blocks
paired together but still separate
to form all kinds of system structure
...
client role
server role
uses

Figure 12 Principles of client-server architecture

As indicated in the right hand side of the diagram, a building-block may be both user and provider of services. Therefore, it may have client and server roles and may participate in many client-server relationships with other building-blocks. It is client or server only in the context of the particular relationship considered.

The realisation of client-server architecture in software is via programming languages and middleware (not shown in figure 12). The physical realisation of client-server architecture consists of networks of separate computers; consequently the term client-server tends to become a synonym for distributed processing.

Client-server architecture is only incidentally about PCs, or use of any other particular kind of technology. However, in current circumstances, it is usually appropriate that client-server is viewed mainly in terms of exploiting PC technology (as in the Gartner definition which we started with in 1.2 above).

This general form of client-server architecture (autonomous building-blocks, client-server relationships, client role, server role) is a fundamental ingredient of OPEN*framework* application architecture.

4 Summary

This paper has described client-server concepts, technology and architecture. Distinctions have been drawn between basic client-server and various formulations beyond it; these are both special cases of general client-server architecture.

The main conclusion is that client-server is fundamental to systems integration: it is *the* way to synthesise large-scale systems from constituent parts. That is why it is of lasting importance.

Acknowledgements

The ideas described in this paper have been stimulated, and in some cases generated, by colleagues in ICL and in collaborations outside ICL: particularly (but not exclusively) Alan Beer, Ron Brunt, Richard Day, Tony Gale, Andrew Herbert, Nic Holt and Mike Threlfall.

References

BRENNER, J.B. *OPENframework* Distributed Application Services. Prentice Hall International (UK) 1993. ISBN 0-13-630518-0.

BRUNT, R.F. An Introduction to OPEN*framework*. *ICL Tech J.* 8(3) pp351-364, 1993.

HAMMER, M. and CHAMPY, J. Reengineering the Corporation: a manifesto for business revolution Harper Collins Publishers, Inc. USA. ISBN 1 85788 029 3.

DUXBURY, P. Legacy Systems in client-server networks: A gateway employing scripted terminal emulation, *ICL Tech J.* Vol. 9 Iss. 1 pp. 102-121, 1994.

ARCHER, B. The Management of Client-server Systems, *ICL Tech J*, Vol. 9 Iss. 1 pp. 122-137, 1994.

How ICL Corporate Systems support Client-server: an Architectural Overview

Richard Day
Corporate Systems, ICL, Manchester, UK

Abstract

This paper provides an overview of the client-server architecture supported by ICL's Corporate Systems. The architecture of Open VME™ is such that the support for client-server working has been a small incremental step in terms of changes to the system, but it provides a significant step forward for users.

The purpose of the architecture is to provide ease of use for the end-user together with secure access to shared corporate data. It does this through showing how to organise systems so that all kinds of platforms can be used for what they do best.

Corporate Servers have evolved over many years to excel at data-intensive processing and TP. This continues to be a major focus, with a new emphasis on client-server TP with maximum exploitation of the PC revolution and the benefits it brings to users.

Other themes include the enabling of immediate Management Information System [MIS] access to all of the enterprise's information and the protection of enterprise information by the provision of frontware for existing corporate applications.

1 Introduction

1.1. Scope of this paper

The scope of this paper is to provide an overview of the client-server architecture supported by ICL's Corporate Client-Server Systems. The architecture [Day, 1993] is a specialisation of the OPEN*framework* systems architecture [Brunt, 1993]. For an overall introduction to client-server, readers should refer to the paper [Brenner, 1994] in this issue of Ingenuity, before reading this paper.

1.2 What is client-server?
An industry view is that client-server is the splitting of an application into tasks that are performed on separate computers, one of which is a programmable workstation (e.g. a PC). However, the reality is more complex than this.

The term *client-server* is used to mean an architecture of distributed processing with well-defined roles for two types of components - clients and servers. The *client* is the driving or initiating component, delegating tasks to a server and making requests of a server, usually for which it awaits a response. A *server* is a component acting on behalf of a client for a well-defined set of functions, e.g. database requests.

The term *client-server* embraces both hierarchic (workstation/server) and co-operative processing (peer-to-peer) dimensions of distributed computing.

1.3 What are ICL's Corporate Client-Server Systems?
A Corporate Client-Server System may include a variety of platforms (e.g. PCs). It always includes an Open VME system or *GOLDRUSH*™ *Mega*SERVER. Throughout this paper the term "Corporate Server" means an Open VME system and/or *GOLDRUSH* server.

"Open VME" is the name of the system which has been developed from VME. It includes all the environments originally supplied by VME (such as TPMS).

1.4 Purpose
Client-server working is a step on the way forward for Corporate Systems - a small step for Open VME, but a big step forward for users.

A Corporate Client-server architecture is the way to provide ease of use for the end-user together with secure access to shared corporate data.

End-user qualities <—————>	**Corporate qualities**
Allows graphical user interface	Uniform, full, secure data access
Improved functionality	Reliable processing
Improved productivity	Scaleable, flexible environment
Improved quality	High throughput and availability

Corporate Client-Server Systems are primarily concerned with business operations which require co-ordination across substantial parts of the enterprise and, increasingly, across external enterprises. Therefore the servers in such systems support mission-critical data and services used by large numbers of clients.

Corporate Servers have evolved over many years to excel at data-intensive processing and TP. They still do.

During the last decade, new types of machine have evolved to excel at other kinds of task, for example:

- graphics workstations and PCs excel at GUI
- PCs and RISC servers excel at MIPS-intensive processing
- high throughput parallel machines will excel as relational database servers.

A corporate client-server architecture is the way of organising systems so that all kinds of platforms can be used for what they do best.

A major focus for Corporate Servers continues to be on TP. The new emphasis is on client-server TP, with maximum exploitation of the PC revolution and the benefits it brings to users. Corporate client-server working enriches this by bringing TP to the desktop.

The architecture of Corporate Client-Server Systems:

- provides a framework for analysing and designing such systems
- identifies the strengths of such systems and how they can best be exploited in a particular context
- provides a basis which guides the development of an integrated product set which meets customers' needs
- provides a structure which gives an understanding of how those products work together and what their purpose is
- defines a basis for interworking and co-existence with other systems with different architectures
- shows how the architecture is constructed
- locates the standards which are becoming important.

2 Trends in Client-server

For a general discussion see the introduction to client-server [Brenner, 1994] in this issue of Ingenuity.

2.1 Business Trends

"By 1995, client-server will be a synonym for computing." [Gartner, 1992].

The percentage of new, multi-user on-line applications that employ a client-server architecture is expected to grow to 75 per cent by 1996 and 90 per cent by the year 2000. Indeed it is hard to find a large corporation that is not either actively developing client-server applications or considering a strategy to do so.

Many businesses are now seeking to re-partition their information systems: businesses increasingly operate in global markets, and involve more organisations in more places.

2.2 Technology Trends

The driving forces include:

- the need for information technology which can support organisational change
- demand for easier-to-use interfaces
- improved price/performance of desktop platforms
- right-sizing: introduction of workstations but with continuing centralisation of shared data and associated processing onto high-integrity server platforms
- demand for more access to decision data and support for business work processes
- existence of high-productivity tools for client-server applications development
- the appearance of parallel servers which support a relational database with high throughput
- increasing realisation of the value of data to business, and the need for corporate qualities, e.g. reliability, security, availability.

2.3 Benefits

The benefits of corporate client-server working include:

- improved functionality, ease of use, productivity and quality for end-users via graphical user interfaces
- more efficient use of computer resources, lowering overall costs

together with:

- uniform, secure, full, easy-to-use, access to shared data and services.

3 Aims of the Architecture

3.1 Overall Aim

The overall aim of the architecture is to provide a unified framework which can accommodate all the common forms of distributed processing. In particular this includes:

- interworking between
  - o all types of desktop applications, and packages from word processors and spreadsheets through to relational toolsets, and
  - o all types of application and data services on Corporate Servers

- interworking with
    - other systems, including those with different architectures.

The following list illustrates typical facilities on the desktop which will interwork in various ways with key services on the Corporate Server.

Desktop	**Corporate Server**
Icons, objects	Files
Mail	X.400, X.500
Workbenches, CASE tools	Dictionary
Word processors, charts, spreadsheets	Relational database
Relational tool-sets	Other database
Presentational applications	Existing applications
Distributed applications	Distributed TP applications Distributed applications

The following list illustrates typical facilities on a Corporate Client-Server System which will interwork in various ways with services on other servers which are either Corporate Servers or servers with a different architecture.

Corporate Client-Server system	**Other systems**
Distributed TP applications	Distributed TP applications
Desktop applications	Relational database
Desktop packages	Dictionaries
Relational database	Existing applications
Other database	X.400, X.500
Dictionary	Distributed applications
Workbenches, CASE tools	Other database
Files	Files
X.400, X.500	
Distributed applications	

A key plank in the architecture is the supply of middleware to enable client-server working. This middleware supports key standards (e.g. SQL standards) and hides fluid interfaces from applications.

3.2 Major Themes

The major themes which run through the architecture are:

- client-server TP: bringing transaction processing to the desktop
- protection of enterprise assets: provision of Frontware for existing corporate applications
- enabling of immediate MIS access to enterprise information
- bringing the Corporate Server to the desktop: access to server objects through icons
- generic support for client-server tools usage
- enabling of co-operative (peer-to-peer) processing between ICL Corporate Client-Server Systems and other systems with the same or a different architecture
- provision of application development tools (including full CASE tools) for generating client-server applications embracing a range of platforms
- ability to manage Corporate Client-Server Systems as a whole and in conjunction with systems with different architectures
- provision of high-throughput parallel servers

3.3 Principles

3.3.1 Openness

The Architecture is an *open* architecture - it provides for:

- any other platforms to be connected to Corporate Servers at the networking level
- the use of any user interface - whatever is appropriate for the application. Graphical user interfaces are supplied by software on the desktop
- application development tools which will generate client-server applications which will run in many different environments. It allows a free choice of application development tools
- application interworking, application portability, tool interworking and avoidance of lock-in
- open SQL access from all clients to all Corporate Server data sources
- interworking with TP applications on other systems
- management of Corporate Client-Server Systems in conjunction with systems with different architectures.

3.3.2 Systems evolution

A key property of modern architectures is their ability to allow for smooth evolution. This allows the enterprise to respond to changing business needs through introducing new technology, evolving applications architectures and accessing corporate information in new ways.

Forwards compatibility
The architecture provides for unchanged applications to be revitalised through the use of frontware, for existing applications to interwork with new applications, for existing data to be accessed in new ways (relational access) and for the evolution of existing systems to incorporate new facilities.

Optimal usage
The architecture supports the use of the various platform elements within the total system to best effect, e.g. graphical user interfaces on PCs and shared corporate data and services on Corporate Servers.

Protection of assets
The enterprise's valuable information resources are protected, including defending against accidental or malicious threats. A guaranteed forwards development path is provided such that investment in application development and stored information is protected.

The architecture provides for server applications to be insulated from ever-changing PC platforms, environments and software package versions.

Corporate Server environments provide for controlled evolution. Applications do not have to be updated when new Corporate Server platforms, environments and facilities appear, since these are supplied in a forwards-compatible way. A particular example is that migration of data to a *GOLDRUSH Mega*SERVER can be achieved without any change being required to relational applications.

3.4 Data access

The architecture:

- supports the need for desktop access to corporate data, revitalisation of existing corporate applications and relational access to data
- provides for access to shared corporate data, taking account of the needs for security, the size of the data and the size of the application components needed to access it. Corporate Servers provide access to databases, files, dictionaries and directories and provide transaction processing and other processing facilities.
- provides for immediate MIS access to enterprise information
- facilitates the minimisation of network traffic in a way consistent with the needs of the application.

3.5 Standards
There are many and diverse standards (international, industry, de facto) that the architecture supports.

The networking element of the architecture enables support of *all* networking standards, by the use of gateways where appropriate.

The application development element of the architecture insulates applications from changing standards in the areas of distributed application services, networking and platforms by removing the need for specific programming to such interfaces.

3.6 Quality
Corporate Client-Server Systems provide corporate levels of integrity, availability, performance, ease of use, management and adaptability.

4 Corporate Client-server Architecture

This description follows on from the paper [Brenner, 1994] in this issue of Ingenuity.

4.1 Models of distributed applications
In fundamental terms, a total application can be thought of as being made up of three parts, as shown in figure 1:

- data management
- application logic
- user interface or presentation

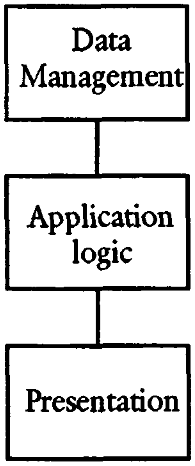


Figure 1 An application divided into three parts

This separation enables the application components to be matched to the system that best suits those functions. For example, the client platform that best matches the user interface is often the PC with its superior GUI capability.

4.1.1 Basic Client-server Models

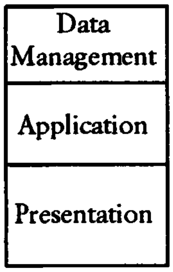


Figure 2 A Monolithic Application

Figure 2 represents a monolithic application, i.e. one that has no client-server split and uses dumb terminals. An application can be split in various ways:

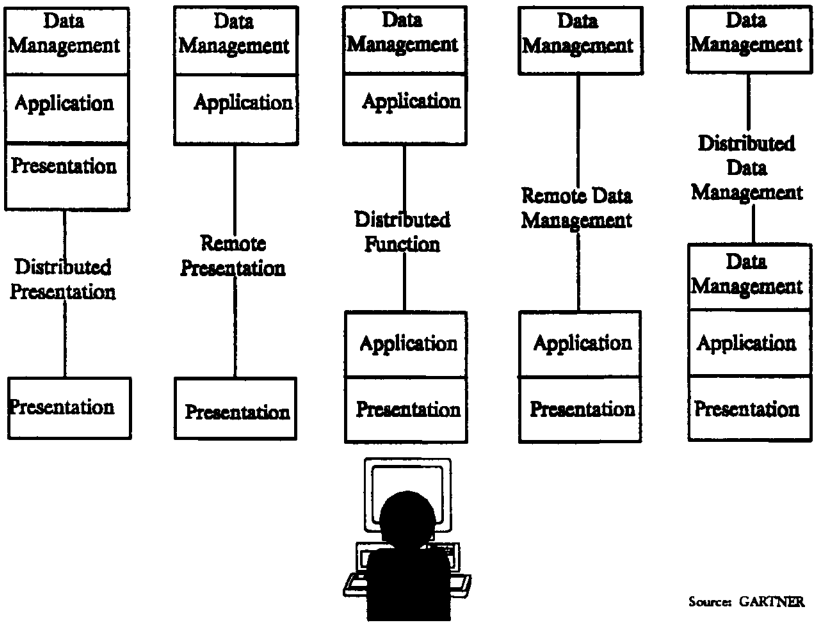


Figure 3 Client-server models

This client-server split (see figure 3) divides the three areas of functionality into two locations separated by a network.

The term *remote* means *remote from the point of view of the application,* and *distributed* refers to *distribution within a given function* (presentation, application or data management).

4.1.2 Co-operative Processing

Figure 4 represents co-operative processing between two server platforms.

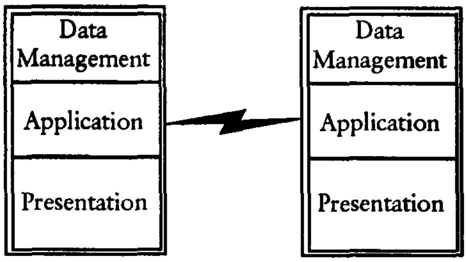


Figure 4 Co-operative Processing

4.1.3 The possibilities combined

In diagrams 3 and 4, due to Gartner, the different distribution models are represented separately. However, an application may conform to all the models at once, as shown in figure 5.

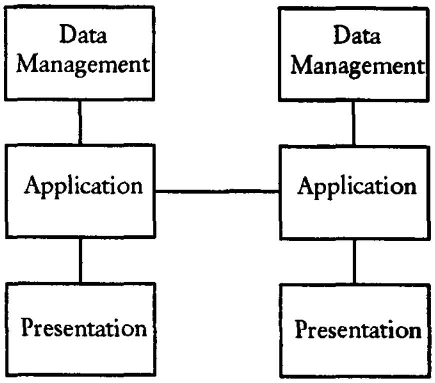


Figure 5 The possibilities combined

In other words we have a unifying architecture which allows a given application to be distributed in a complex way, where required, and which allows different applications to be constructed in different ways whilst, where required, accessing the same data.

Application development tools are used to generate the application, and systems management tools manage systems and objects within them.

Figure 6 represents an application, together with these services and tools. The lines between the boxes represent places where there is potential for distribution. Where more than one application, presentation or data management component is on the same platform then the corresponding boxes may be amalgamated.

It is possible for an application to be distributed at all the places shown in the diagram.

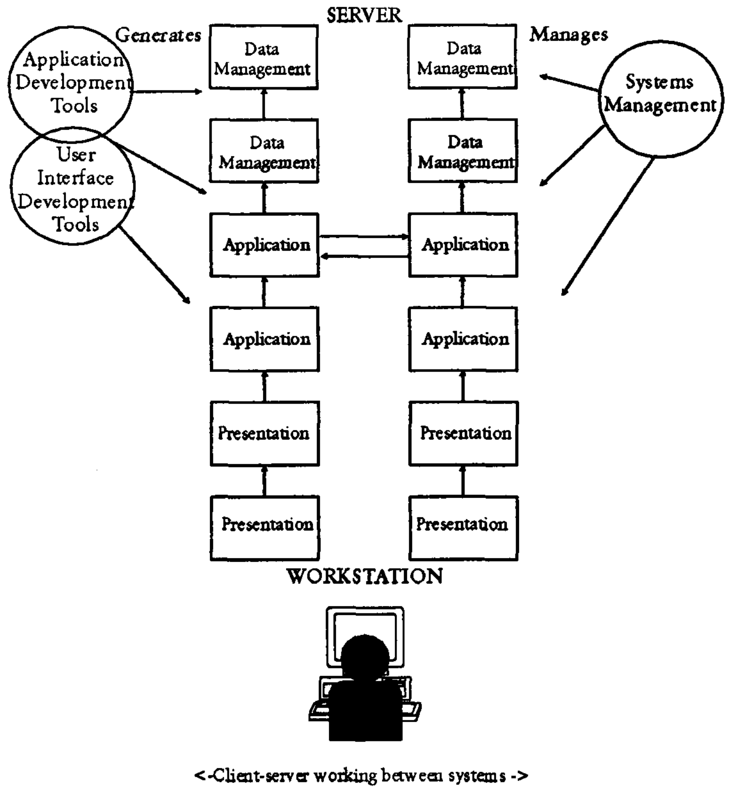


Figure 6 Potential for Distribution of functions

In figure 6, the vertical dimension represents client-server working from workstations through to servers; the horizontal, client-server working between systems - *co-operative processing*.

The architecture provides for generic middleware, to enable full functioning of applications across the network connections, through key enabling technology (see figure 7).

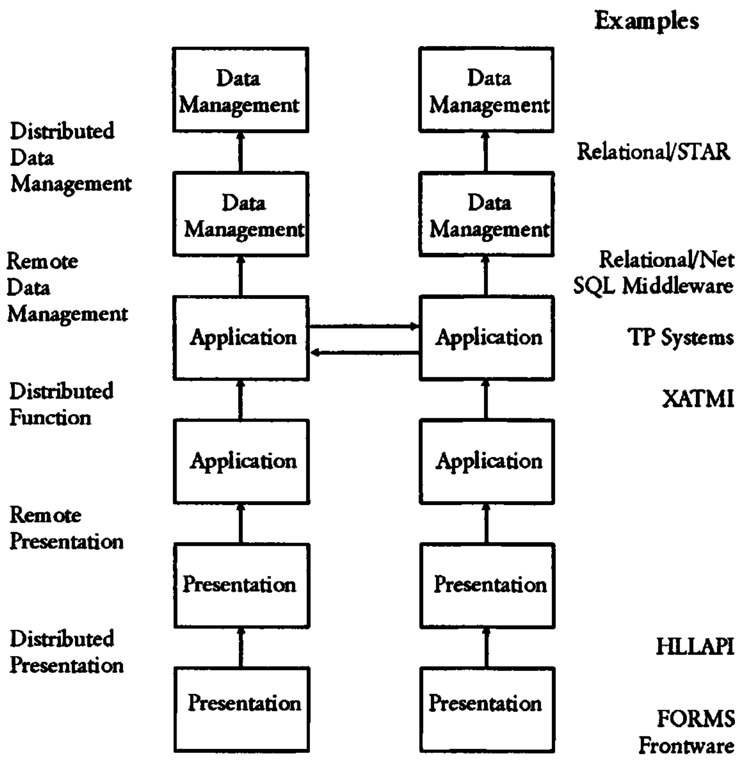


Figure 7 The Unified Corporate Client-Server Architecture

4.2 Models - Summary

This section contains a short description of each of the models. The most significant four models are described first.

4.2.1 Distributed Presentation

In this model, the presentation services are distributed between the workstation and a host. An example of this is the FORMS product. This model also supports the use of Frontware to improve the presentation of unchanged Corporate Server applications.

4.2.2 Distributed Function

In this model, the application functions are distributed between the client and the server. This type of usage will grow significantly as enabling infrastructure and development tools become more widely available.

Transaction management systems are major contributors to the support of this model.

4.2.3 Remote Data Management

In this model, the whole of the application is at the client end and the data management services are on the shared server. An example of this is the usage of relational networking products to enable MIS access to shared databases. This type of usage is growing rapidly now.

4.2.4 Co-operative Processing - application to application

In this model, applications communicate on a peer-to-peer (equal) basis. Where access to several databases on possibly different systems is required, then this is an appropriate model to use.

The benefits of this model are in high throughput access to data, with co-ordinated update to data where transaction management facilities are used.

4.2.5 Distributed Data Management

In this model (otherwise known as distributed database) access to several databases is achieved through database management system technology. An example of this is the usage of relational technology such as INGRES/STAR*.

Benefits can be derived from using this model, but in many circumstances more benefits are achieved from the co-operative processing (application to application) model which we expect to be more widely used.

4.2.6 Remote Presentation

In this model, the presentation services are on the workstation and are driven from the application on the server. An example of this is X/Windows driven remotely. This model is not expected to become a dominant architecture in a corporate systems context.

4.3 Specialised Example

The architecture may be specialised in various ways.

The following example (figure 8) is taken from a Financial Services enterprise's architecture. It shows combined use of distributed function, co-operative processing and remote data management.

5 Models - description

This section further describes each of these client-server models in turn, together with their benefits. It relates the architectural aims to the architecture and indicates how it is supported through an enabling product. The most significant models are described first.

5.1 Distributed Presentation Model

> **A major theme of the architecture, expressed through this model (see figure 9), is to supply Frontware Support for existing (and new) Corporate Server applications.**

The provision of frontware support allows existing Open VME applications which use terminals to be given interfaces which are more user friendly, without change to the applications. It is also applicable to new applications.

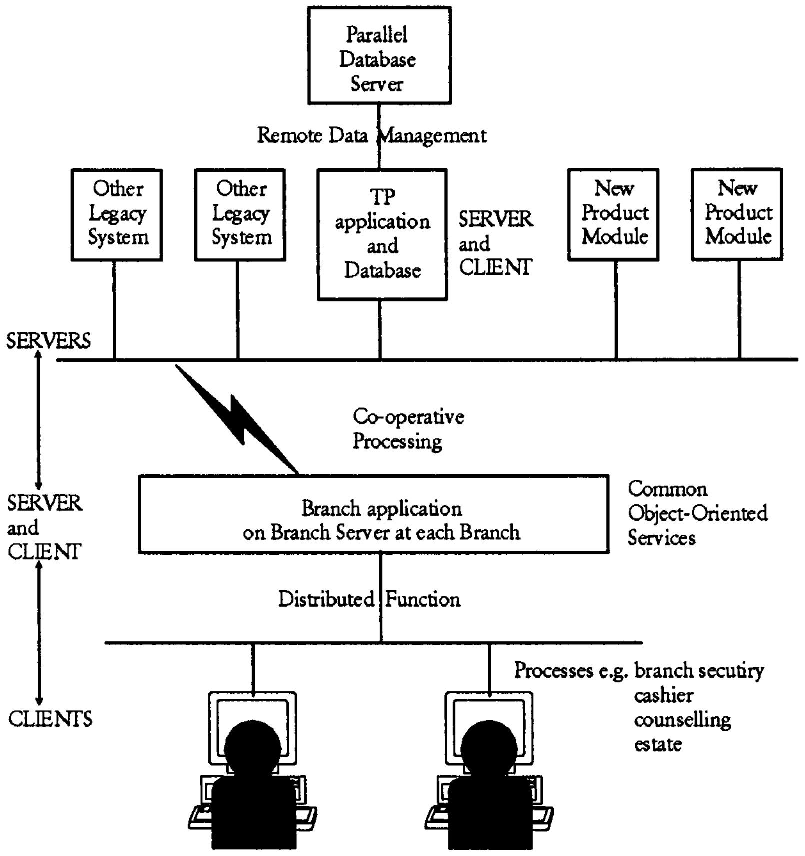
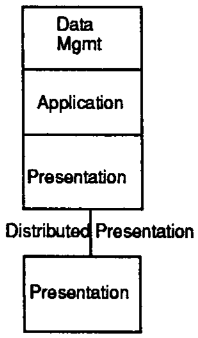


Figure 8 Financial Services Generic Model



- **Distributed Presentation Services**
- **Key usage: *re-vitalise existing Corporate Server applications without having to change them***
- **Example: automatic generation, with FORMS - Windows front end to VME applications**
- **Generic Frontware Enabling: allows free choice of graphical toolsets**

Figure 9 Distributed Presentation

The architecture provides a single, integrated approach to this so that:

- as an optional first step, applications may be *automatically* updated to have a Windows interface.
- over a period of time the same applications may have a PC application added to them, without changing the existing VME applications in any way.

This integrated approach is achieved through using FORMS together with toolkits such as Microsoft® Visual Basic.

The two stages of this approach (which can be mixed in any way) are now described.

5.1.1 Automatic generation

This stage allows the automatic updating of TPMS applications which use terminals so that they can have a user interface running under Windows. It is equally suitable for new applications.

Characteristics:

- There is no user application on the PC.
- Existing applications may be re-engineered automatically to use the new user interface.
- End users work in a Windows environment with facilities such as mouse, pull-down menus, pushbuttons, radio buttons, drop-down lists and cut-and-paste.
- A text-based form-filling interface is provided.

This approach is supported by ICL's FORMS product. The user interface definitions are generated using the Data Dictionary.

5.1.2 Flexible, controlled integration of frontware

This stage allows a further improved graphical user interface and business logic on the PC to be added to an unchanged TPMS application, with or without integration with FORMS.

- It may be used with applications which were written to use terminals.
- It may be used with applications which use FORMS, as a replacement for the FORMS presentation, or in combination with it.

An improved graphical user interface may be constructed using toolkits such as Microsoft Visual Basic, Visual C++ or Gupta SQL Windows.

Dialogue Manager is also supported. This acts as a front-end to both new and existing TP systems including TUXEDO® on UNIX®, TPMS and MVS CICS.

5.1.3 Generic Frontware Enabling

The architecture provides generic infrastructure, which will allow unchanged TPMS applications to interwork with graphical and other applications on the PC.

Characteristics:

- A wide choice of tools which generate frontware is provided for.
- A workstation application is generated.
- The generated frontware may be very flexible: graphical objects may be displayed; large amounts of processing may be carried out at the workstation; information may be accumulated from a number of different VME application outputs.
- A consistent image can be given across VME applications and applications running on other vendors' mainframes.

By providing an interface on the PC (3270 DLL) which maps onto the ICL terminal protocol (7561), then all those frontware packages (e.g. Easel, Mozart, QuickFront, Flashpoint, Micro Focus Dialogue System DS/3270) which can generate client applications which work with unchanged CICS applications may be used to generate applications which work with unchanged TPMS applications.

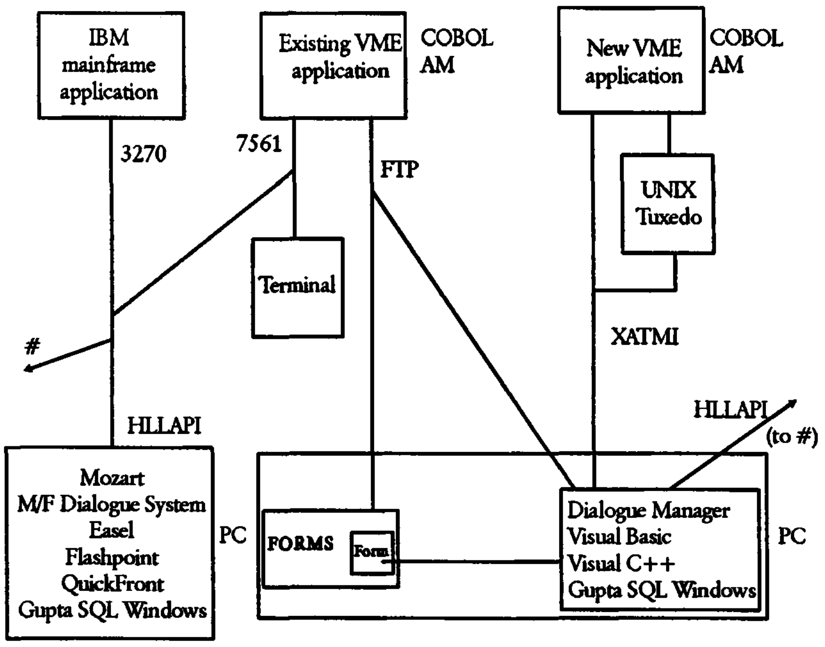


Figure 10 Frontware Possibilities

Further remarks

The relevant IBM standard supported is HLLAPI. An enabling PC product (HLLAPI/7561) will be available shortly from ICL.

Similar infrastructure is available on UNIX, and is applicable for any client platform.

Since an application exists on the PC (which can have any degree of complexity required), this frontware usage should perhaps be considered as a distributed function conforming to the model.

Figure 10 shows the various possibilities for frontware.

5.2 Distributed Function Model

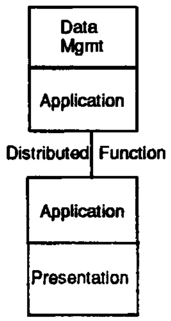
A major theme of the architecture, expressed through this model (see figure 11), is to extend Transaction Processing to the desktop.

A secondary theme is to enable non-TP client-server access, e.g. for development tools and workflow systems.

The major benefits of this model are that any combination of user interface styles can be used; the client part of the application is insulated from knowledge of particular data structures; the server part of the application is insulated from knowledge of particular user interface style, and high throughput systems can be supported safely and securely.

Where a large application is needed to support critical business functions, in many cases it is not appropriate to run it totally on PCs.

Firstly, it is often inappropriate to distribute data. For example, in the world of financial services, customers need information about their accounts from any ATM and financial institutions need access to information associated with customers. These needs are most easily met by having a shared database on a server platform. Distributing data may result in the introduction of many complexities into applications, without any corresponding benefit being obtained.



- **Application function distributed between client and server**
- **Key usage: *bringing TP to the desktop***
- **Example: Open AM programme for generating such applications**
- **Generic middleware for client/server TP support**
- **Benefits: separation of concerns, shared data, high throughput**

Figure 11 Distributed Function

Secondly, the part of the application that is accessing this data is best run on a server platform. This gives an appropriate level of security to database access, and, through the use of transaction management facilities, a controlled environment is provided, giving scheduling, resilience and co-ordinated update to data. Compared with the remote data management model where SQL is being passed, network traffic from PCs can be reduced, giving better throughput.

5.2.1 Client-server TP

The architecture provides generic infrastructure which supports PC-client-server TP applications with function distributed between the PC and the Corporate Server or other server.

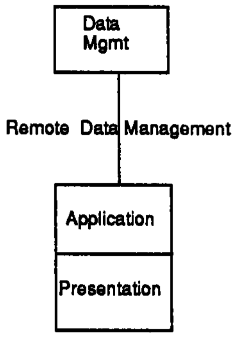
The intention is to allow any client application on the PC, whether written or generated, to interwork with a server application on TPMS. Transaction semantics will be fully supported. More than one interaction per TP phase will be supported. The X/Open standard application interfaces (initially XATMI, followed by CPI-C and TxRPC) will be supported on Open VME and at the PC. Client applications (e.g. Dialogue Manager ones) which work with any other X/Open conformant transaction management system, e.g. TUXEDO, should work unchanged with TPMS server applications.

5.2.2 Generic Infrastructure

The architecture provides for generic infrastructure to support distributed function applications which may not require TP facilities (e.g. application development tools). This will be supplied through the Open VME sponsor and gateway architecture, by the support of X/Open interfaces (as above) on PCs and on Open VME. Remote Procedure Call (RPC) interfaces will be included.

Support for Electronic Data Interchange (EDI) based applications is also provided.

5.3 Remote Data Management Model



- **Application all at client end, with shared data**
- **Benefits: *Immediate MIS access from all PC applications to all corporate data***
- **Example: SQL access to all Corporate Server data sources (via ReView)**

Figure 12 Remote Data Management

A major theme of the architecture, expressed through this model (see figure 12), is to enable immediate MIS access to enterprise information.

The principal standards relating to this model are SQL standards. Open SQL access to all Corporate data sources is provided.

A second theme is to provide access to *GOLDRUSH* databases.

For high-throughput access to databases, the usage of this model should be combined with the usage of a transaction management system.

5.3.1 Open SQL Access

A major characteristic of SQL is that it allows queries to be made which are not pre-determined. This is extremely powerful: if the required information is modelled in the database then it can be extracted, immediately, without the need for any user application.

SQL is now a standard supported extremely widely by desktop applications and packages. We support SQL access to all Corporate Server data sources. Therefore we can enable information to flow between the server and the desktop.

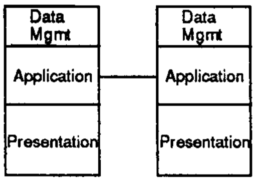
The architecture provides generic infrastructure and product which allows:

- relational front-end tool sets
- graphical 4GLs for client-server development (such as Gupta's SQLWindows and Powersoft's PowerBuilder)
- other packages such as spreadsheets

to interwork with all Corporate Server data sources, including relational databases, IDMSX databases and files including indexed sequential files.

This is enabled by the ReView product.

5.4 Co-operative Processing (application to application) Model



- **Applications communicate on a peer-to-peer (equal) basis**
- **Benefits: *co-operative processing*: high-throughput to data, where there are several databases which may be on different systems; with co-ordinated update (via TP). Separation of concerns**
- **Example: implemented through TPMS Open TP facilities**

Figure 13 Co-operative Processing

A major theme of the architecture, expressed through this model (see figure 13), is to enable co-operative (peer-to-peer) processing between Corporate Client-Server Systems and other systems with the same or a different architecture.

The benefits of this model are in high throughput access to data where there are several databases which may be on different systems. Co-ordinated update to these databases, which may be accessed through different database management systems, is achieved through the use of transaction management facilities.

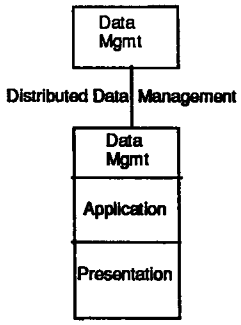
An application component is associated with a given database and does not need to have any knowledge of the data types and layouts in other databases. These application/database components could, therefore, be considered as large scale objects.

The application components may be under separate ownership and operational control, have different technical origins and use different infrastructures.

This usage will be supported by Open VME TPMS facilities through OSI-TP and X/Open interfaces, including XATMI, (and by similar facilities in TUXEDO).

Support for EDI based applications is also provided.

5.5 Distributed Data Management Model



- **Access to several databases achieved through DBMS technology**
- **Unique benefit: queries will find data wherever it is, and combine it.**
- **Often benefits are better achieved through co-operative processing (application to application) model: throughput; separation of concerns**

Figure 14 Distributed Data Management

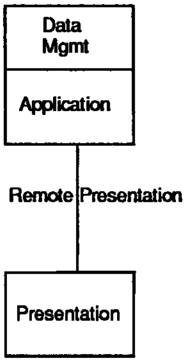
Many of the potential benefits of this model (see figure 14) can be achieved more readily through the co-operative processing (application to application) model, by using distributed transaction processing, for example.

The unique benefit of distributed database technology is that it provides the ability to perform queries that will find the data wherever it is held, if necessary by combining data from several databases. However the performance aspects of this facility mean that often distributed database services will be used for MIS work but distributed transaction processing for applications.

When distributed data management is used by applications, the data types and layouts in the databases need to be known by the applications. Therefore there may be difficulties of management where the application components are under separate ownership or operational control, or have different technical origins or use different infrastructures.

This is exemplified by X/Windows support through client-server PC/UNIX components of the system. Disadvantages of this model (see figure 15) are firstly that the application often has to have knowledge of the details of the presentation and secondly that the network traffic tends to be high where details of graphical bit-mappings are passed.

5.6 Remote Presentation Model



- **Presentation services on workstation, driven from application on server**
- **Exemplified by X/Windows client/server**
- **High network traffic where graphical bit-mappings are passed**

Figure 15 Remote Presentation

6 Architectural Elements provided

Corporate Client-Server Systems are constructed from the following architectural elements:

- user interface
- distributed application services
- transaction management
- information management
- application development
- systems management
- networking services
- platforms.

These elements are briefly outlined in this section, showing how they relate to the Corporate Client-server architecture.

6.1 User Interface

The fundamental principle is that the architecture supports the use of any user interface - whatever is appropriate for the application:

- the architecture allows the use of any platforms and environments which support user presentation. Sources include terminals, PCs, tone phones, videotext terminals, ATMs, multimedia etc.
- character interfaces and graphical user interfaces can be supplied.
- application development tools will generate server applications which will work with any client applications, thereby allowing a free choice of presentation.

6.2 Distributed Application Services

A major difficulty in distributed computing is that many different standards apply, and there is uncertainty about which to use. There are different kinds of infrastructure: the architecture *embraces* many of these environments and provides for co-existence with the others. It achieves this through:

- provision of application development tools which hide the different interfaces from the applications
- provision of distributed application infrastructure through the use of TPMS components, TPMS developments and the extension of TP to the desktop
- provision of middleware as described, and further middleware as standards progress
- porting of middleware products to appropriate platforms.

6.3 Transaction Management

Transaction-based distributed computing is applicable to mission-critical systems. It supports reliable computing by processing business transactions while preserving the integrity of shared data.

A key feature of the Transaction Management architecture is that a distributed transaction can be implemented with parts that execute on different systems, yet the whole transaction will either succeed and complete, or the whole transaction will be rolled back as if it had never begun.

Transaction Management systems are major architectural contributors to the achieving of corporate client-server computing.

This is achieved through support by such systems of distributed function and co-operative processing. The extension of TP to the desktop has been described in section 5.2.

6.3.1 Aims

The architecture provides for a transaction management environment which conforms to the following principles:

Quality: To support high-throughput mission-critical systems reliably.

Openness: To support:

- applications interworking with applications running on other systems
- the porting of applications and packages from and to other Transaction Management systems
- the use of open interfaces in combination with many current interfaces
- client-server access, including access from the desktop

through support of the X/Open TP model.

Forwards Compatibility:

To allow:

- existing Open VME TP applications to be enhanced to use open interfaces
- existing Open VME TP relational applications to exploit *GOLDRUSH*, without change.

Enterprises' investment in TPMS applications is protected through TPMS being enhanced to meet these transaction management aims.

6.4 Information Management

Information is a valuable commodity. A key feature of the Information Management Architecture is that the value of an enterprise's investment in information is maximised, by safeguarding it, by allowing it to become available throughout the enterprise where required, and by providing for open client-server access to it.

6.4.1 Aims

The Information Management element of the architecture conforms to the following principles:

Data Access

Provision of:

- high-throughput access to shared corporate data, taking account of the needs of security and the size of the data
- support for relational database, Codasyl database (IDMSX) and files (including sequential and indexed).

Immediate MIS-type access to enterprise information

Openness

Provision of open SQL access from all clients to all Corporate Server data sources.

6.4.2 Client-server SQL Access to All Corporate Server Data Sources

SQL access to all Corporate Server data sources is supplied through the ReView product (see figure 16).

Full read access is available today, treating these views as though they were relational tables.

This full read access extends to interactive SQL, SQL embedded in C and COBOL, all INGRES tools and all database-independent tools which interface to INGRES (potentially ICL TeamTOOLS, ACCELL, Gupta, FOCUS, TechGnosis, etc.) and any other vendor's tools which access INGRES databases.

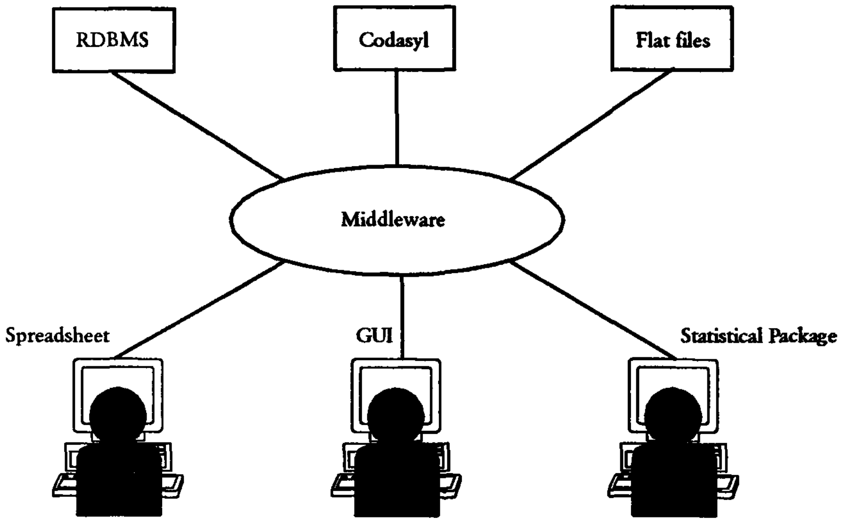


Figure 16 Access to all corporate data sources from most PC packages and applications

The applications and tools may be run on any environment on which they are supported and which can network through to Open VME using INGRES/NET - PCs, UNIX platforms and all the Open VME environments (VME-X, TPMS, MAC, Batch).

ReView gives full SQL read access to:

- many IDMSX databases as though they were one relational database
- many ISAM files as though they were one relational database

- many sequential files as though they were one relational database
- any combination of the above
- any combination of the above together with one INGRES database.

Additionally, it is possible to access the Open VME catalogue, operator picture files, X.500 directories, etc., etc.

Access through CAFS™ is possible.

Remote Data Management via SQL is supported, i.e. the front end usage does not need to be on the same platform as the databases and may be on UNIX or PCs.

The architecture provides for:

- *performance:* supply of ReView on *GOLDRUSH* (accessing IDMSX, etc. on Open VME and INGRES on *GOLDRUSH*)
- *automation:* simplification of view declaration, automatic registration process, automation from DDS business data
- *update facilities:* these can be provided, although update cannot be done in a fully automated way through standard SQL.

6.4.3 Access to all corporate data from all types of desktop applications and packages

There is a choice of interworking mechanisms:

One method of achieving networking with relational tool-sets is to use the relational/NET products.

The architecture provides for the supply on Open VME systems, *GOLDRUSH* servers and UNIX systems of product which achieves interworking from applications and many packages such as spreadsheets.

Examples of such middleware products include TechGnosis's SequeLink, and Information Builders' EDA/SQL.

Products such as these connect:

- *from* PC products such as Visual Basic, Lotus 1-2-3, WingZ®, Hypercard, Easel, ICL TeamTOOLS, SQL, Windows and many others
- *to* many database management systems on many server environments such as Open VME, UNIX, MVS and others

providing access to all ICL Corporate Server data sources via ReView.

Sequelink server has been ported to Open VME.

6.4.4 Relational Database

A choice of relational database management systems is provided: an SQL service will be supplied by INGRES and Oracle on *GOLDRUSH* and by INGRES, Oracle and Informix® on Open VME.

6.4.5 Codasyl Database - IDMSX

IDMSX is fundamental to our architecture and will continue to be supported and developed to exploit the increased power and size of new machines.

6.4.6 Application Development

The overall aim of the application development element is to enable application developers to create applications needed to support an enterprise information system.

The architecture provides a client-server application development environment. Development workbenches operate at the desktop, in conjunction with shared master information about business processes and data and about computer processes and data held on the Corporate Server in a corporate dictionary.

6.4.7 Data Dictionary System

Quality

DDS is well-placed to be the central repository for corporate business dictionary data - providing high functionality and integration with many CASE tools.

For corporations looking for a single place to retain their corporate data, and where they have different CASE tools on different systems, DDS is ideal.

Openness and client-server access

Not only is QuickBuild heavily integrated with it, but there are, and will be, links between DDS and leading CASE toolsets and UNIX dictionary products from leading vendors. Additional CASE tools will be supported through support of the CASE Data Interchange Format (CDIF) standard.

- At first release, the DDS CDIF product ("DDS Data Interchange") is capable of importing and exporting all the subject areas defined in the Interim CDIF standard - Entity-Relationship-Attribute diagrams, Data Flow Diagrams and Data Inventories - as well as all DDS elements and properties in CDIF format using ICL-specific extensions.
- The product is issued with an associated service, so that users can be advised on the capabilities of the product and how those capabilities can be tailored.

An object-oriented view of DDS objects is supported through the PC-based browser/editor DDS WINDOWS product (which is available today) providing the following key features:

- graphical user interface to DDS, in a Microsoft Windows environment
- navigation through DDS
- browsing edit of DDS where the changes are prepared locally and a single update is issued to apply the changes.

The architecture provides for:

- description in DDS of more types of object, e.g. real-world objects, program text, graphics objects, C programs and data
- improved ability to generate for IBM and UNIX systems
- re-map of DDS to use relational rather than IDMSX, giving potential to port to *GOLDRUSH.*

6.5 Systems Management

The architecture's Systems Management element aims to manage Corporate Client-Server Systems as a whole, and to manage them in conjunction with systems with different architectures.

6.6 Networking Services

The aim of the architecture, at the networking level, is to provide interworking between our Corporate Servers and any other platforms.

This is achieved through a combination of direct support within the Corporate Server for a set of core networking services, and provision of gateway products.

6.7 Platforms

The architecture provides for the different application components to be hosted on various appropriate platforms to best effect.

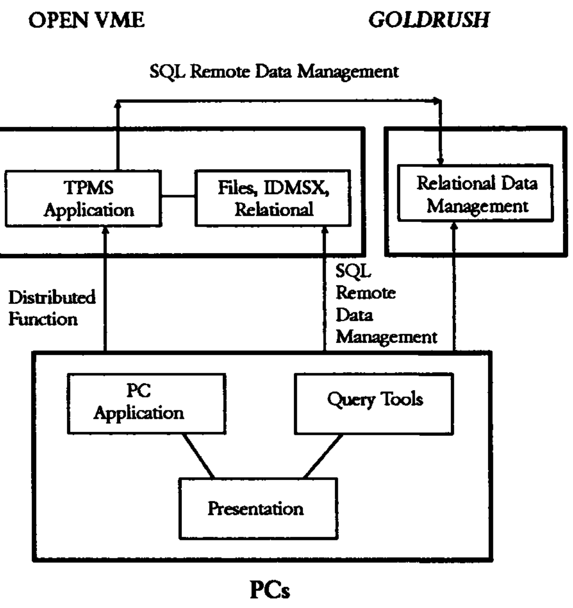


Figure 17 Platform usage

Open VME systems will continue to be enhanced by a range of faster systems supporting the same order code. *GOLDRUSH* enables an order of magnitude increase in relational database performance, together with all the corporate qualities, without change being required to applications.

In a distributed function situation, server code runs on Open VME. This code will normally be running under TPMS, obtaining its benefits as described in other sections. Figure 17 shows a typical arrangement, with relational access to *GOLDRUSH*. Application access to data management on both Open VME systems and *GOLDRUSH* servers is via TPMS; remote data management access (MIS access) to both Open VME and *GOLDRUSH* is via SQL. Data management access from TPMS to relational databases on Open VME and *GOLDRUSH* is via SQL.

7 Summary

Enterprises' application architectures, to be effective, need to use various computing platforms together, each for what they do best. In order to provide ease of use to the end-user, programmable workstations, such as PCs, are likely to appear in the architecture. In order to provide secure access to shared corporate data, Corporate Servers also appear.

This inevitably gives rise to a client-server architecture.

This paper has discussed what client-server means and how ICL's Corporate Systems support it through:

- bringing TP to the desktop
- providing frontware for existing corporate applications
- enabling immediate MIS access to enterprise information
- providing high-throughput database servers (*GOLDRUSH*).

Acknowledgements

TUXEDO and UNIX are registered trademarks of UNIX Systems Laboratories Inc. in the USA and other countries.

X/OPEN is a trademark of X/Open Company Limited in the UK and other countries.

Microsoft is a registered trademark of Microsoft Corporation in the USA and other countries.

Informix and WingZ are registered trademarks of Informix Software Inc. in the USA and other countries.

Ingres is a trademark of the Ingres Product Division of ASK Incorporated.

VME, CAFS, *GOLDRUSH* and QUICK-BUILD are trademarks of International Computers Limited.

Reference in this paper to products other than ICL products is intended solely to illustrate technical principles described in the architecture, and not to assist readers in evaluating or using the products concerned.

References

BRENNER, J.B. Client-server architecture, *ICL Tech. J.* Vol. 9 Iss. 1 pp. 3-17, 1994.

BRUNT, R.F. An Introduction to OPEN*framework*. *ICL Tech. J.* 8(3), pp. 351-364, 1993

DAY, R.P. The Architecture of ICL Corporate Client-Server Systems document. CSD 259 issue 1, Aug 93

GARTNER GROUP INC. Fifth Annual Applications Development & Management Strategies Conference. Gartner Group Inc., 1992.

Biography

Richard Day

Richard Day joined ICL in 1969 with an honours degree in Mathematics from Warwick University, and is now also a Chartered Engineer.

After early work in design, implementation and testing of System 4 COBOL components, he went on to lead the design and implementation of a PL/1 code generator. Involvement in 2900 COBOL compilation systems resulted in his becoming chief designer and then project manager for that development, which was released as VME (C2) COBOL.

Later, he had management and design responsibility running a strategy and design unit responsible for VME application development and database, including language compilers, testing tools, a programmers workbench and VME INGRES.

Recently he was appointed a Systems Architect, with overall responsibility for Corporate Systems' Superstructure (application development and database) and Client-server technical strategy and architecture.

Exploiting Client-server Computing to Meet the Needs of Retail Banking Organisations

Mike Haynes, Geoff Ibbett and Dave Walker
Financial Services Systems, ICL, Slough, UK

Abstract

Retail banking organisations face increasing competition both from other banks and other financial institutions. This trend is being exacerbated by the deregulation of the financial services industry which started in the mid 1980s and is placing increasing pressures on the information technology systems in these organisations. This article describes ways in which ICL uses client-server technology to implement systems capable of coping with the business and technology changes facing the financial services industry.

1 Introduction

1.1 Background

This article describes the approach ICL Financial Services Systems has adopted in developing a client-server architecture addressing the needs of financial services organisations. It also introduces the components from which ICL builds client-server systems for this marketplace.

In late 1990, ICL brought together all of its financial services activities into a new global business. The objective of this unit, now known as Financial Services Systems, was to develop a coherent strategy for the use of information technology (IT) within retail banking organisations in selected key countries and then to develop products and services enabling the concepts that made up this strategy to be delivered to the customers.

Although the structure and organisation of financial institutions vary from country to country, it is commonly accepted that retail banking is a business offering core banking services (that is, deposits, lending and money transmissions) to the general public. However, as financial service organisations come to offer an increasingly wide variety of services, such as pensions and insurance, this definition becomes inadequate.

In the United Kingdom, the main providers of retail banking are high street banks and building societies, although large retail organisations, such as Marks and Spencer, now offer an increasing range of financial products. In other countries, the dominant suppliers of such facilities can include organisations such as the postal savings banks or credit associations.

Throughout this paper, the terms *retail banking* and *bank* are used rather than the more general (and probably more accurate) term *retail financial services.*

1.2 The Retail Banking Environment

In the 1990s retail banks will need to respond rapidly to a business environment characterised by uncertainty and change. Banks are historically conservative and averse to taking risks, and will not meet the challenge of change easily. The need for banks to respond to change is paramount. Only those organisations that learn to adapt to change are likely to survive.

Some of the pressures for change on retail banking organisations are shown in figure 1.

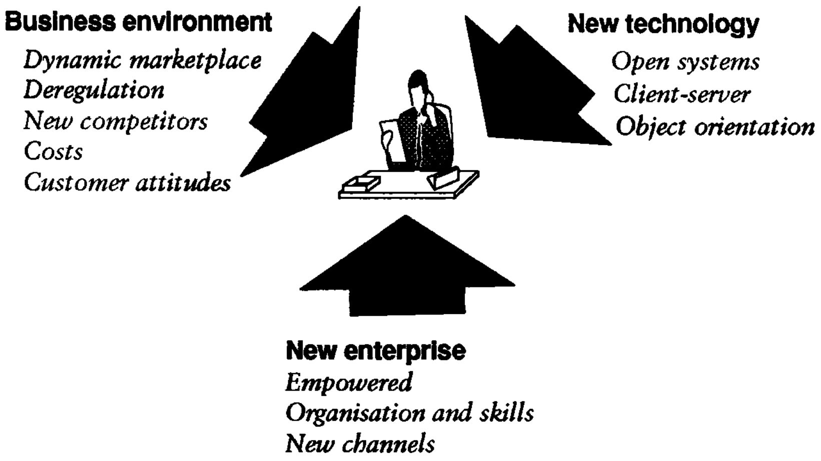


Figure 1 Pressures on Retail Banking Organisations

Business Environment

Intense competition in the marketplace, accelerated by widespread deregulation of the financial services industry, has dramatically changed the business environment, forcing banks to react more rapidly than in the past to changes in the market environment. Retail banking organisations look for ways to reduce staffing levels and restructure their networks of high street branches while simultaneously reprofiling the main staff role from administration towards sales and services.

Customers are increasingly demanding to do business on their terms, using methods like telephone banking and self-service machines, rather than in the traditional ways. Many of the staff reductions in recent years are as a consequence of this change in attitude by customers.

New Enterprise

In banking, the concept of a branch is changing significantly with new forms of distribution channels such as telephone and home banking increasing in popularity and availability, redefining the roles of branch staff and their jobs.

Information technology departments are being asked to deliver systems supporting this much more volatile and dynamic form of business environment.

1.3 New Technology

In an industry in which most products are essentially the same, information technology is seen as potentially a key instrument for reducing costs and delivering competitive advantage.

Thus the key requirement is an ability to deliver flexible services to a wide variety of different types of user, and through a variety of channels.

The rapid advance of technology in both hardware and software terms has dramatically changed costs determining the most effective solutions. This effect is particularly apparent in personal computers and open systems technology.

There is an increasing need for systems which require the integration of components from a wide variety of suppliers. This approach is facilitated by the emergence of both formal and de-facto standards covering all elements of IT solutions.

2 The Problem

2.1 The Existing Infrastructure

Retail banking organisations vary greatly in the way their information technology systems are organised. Figure 2 shows a typical case. The key characteristics are:

- centrally located core systems
- a geographically distributed branch network
- links to third party organisations
- a wide variety of different device types at different locations within the network.

The sheer size of many of these systems and networks creates major issues in the areas of systems management, change control and the logistics of implementing change [Archer, 1994].

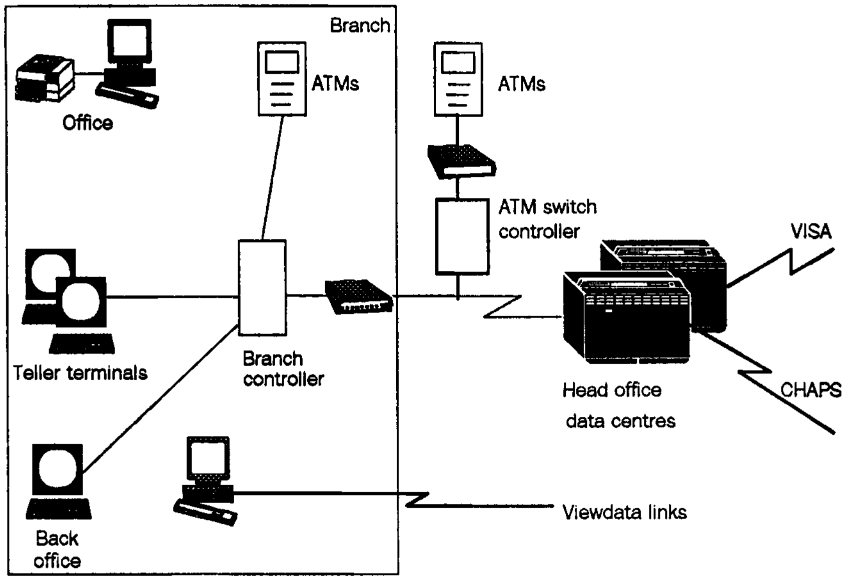


Figure 2 Typical Organisation of IT in Retail Banking

2.2 Requirements

Business Related

For financial services organisations to compete in the dynamic marketplace of the late 1990s they must protect their existing customer base, attract new customers and increase their profitability. Banks can retain existing customers and attract new customers by improving customer service, responding more quickly to customer requests and bringing new products to market quickly. For example they could:

- Provide customer-centred data and systems that give the bank a complete picture of customer's relationships with the bank (in contrast with account-centred systems).
- Develop new distribution channels for delivering existing and new products, for example home banking and sophisticated self-service capabilities.
- Provide *any branch banking*, enabling customers to access from any branch the same facilities that are available at their home branch.
- Increase the flexibility of their existing information systems to make it easier to offer additional facilities with existing products.
- Reduce the time to market of new financial products. The timely introduction of new competitive products stops customers looking elsewhere; that is, it provides *one-stop shopping*.
- Reduce or remove service charges on customer transactions.

Profitability can best be addressed by increasing staff productivity and reducing capital and operational expenditure. For example banks could:

- Provide bank staff with more effective information technology to reduce repetitive tasks and improve decision making.
- Reengineer banking processes to minimise or eliminate redundant activities.
- Reduce capital expenditure and maintenance costs of central, regional and branch information technology
- Target sales and marketing activities more precisely.

Technology and Architectural attributes

The attributes and characteristics of the technology and IT systems that can help retail banks meet these requirements can now be deduced. They include:

- *Consistency.* A consistent technology platform (hardware, operating system, communications, database, etc.) upon which organisations can implement a variety of business functions.
- *Potential for change.* Systems must have the ability to be altered in response to both business and technology changes with the minimum of cost and impact.
- *Rapid development.* Systems must enable banks to bring new financial products to market faster.
- *Integration.* Most financial services organisations have very large, and often complex, core systems characterised by high volumes of database accesses and connections to external services such as card networks and inter-bank clearing houses. These existing systems are the lifeblood of the business. New application systems at branch level must be capable of integrating with these core systems with the minimum of disruption, and also integrate business functionality with standard PC-based application packages, such as electronic mail, word processing and graphics.
- *Flexibility.* Systems must include components that can be used to build solutions which meet the specific requirements of a wide range of organisations, all doing business differently.

It is important that new systems do not compromise the key attributes of existing banking information technology infrastructures namely:

- high branch connectivity
- high throughput on-line transaction processing (OLTP)
- high system availability and security

3 Why Client-Server

Figure 3 illustrates the general concept of relationships between components in a client-server architecture. The server component provides a service to the client.

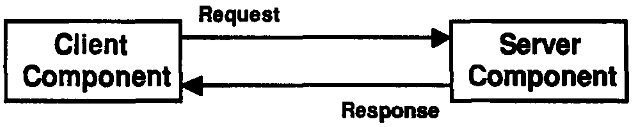


Figure 3 Simple Client-server Model

In reality, client-server interactions tend to be far more complex than is shown in the figure with multiple levels of interaction, all of which are in a client-server relationship to one another. However, a full description of client-server is outside the scope of this article which will concentrate on the specific models supported by ICL's financial services architecture [see papers by Brenner and Day in this issue].

The client-server approach, coupled with object-orientation, provides for simplicity in design, flexibility in implementation and potential for change in response to both business and technology pressures.

The separation of application functionality into client and server components maps very well onto the structure of banking organisations in which branches act as clients and core systems as servers. Provided the interfaces between these components are clearly defined, it is possible to enhance, develop and replace branch and core systems relatively independently of one another.

Encapsulation enables the complexity of existing legacy systems to be hidden behind well-defined interfaces whilst new, client applications are created.

The use of object-oriented techniques to decompose clients and servers further into interacting components enables the development of reusable software objects.

The trend is towards increased distribution of function. The use of client-server object-oriented design and implementation gives clearly defined points at which functions can be distributed. By using implementation technologies that allow clients and servers to interact without knowing each other's physical location, it is much easier to change the ways in which functions are distributed at a later date.

This separation of function means that it is possible to isolate changes to small sections of the total system. This is key in the banking environment where it is vital that change can be implemented with minimal risk. The large scale of many banking systems means that incremental change is also far more practical as it allows it to happen in many small steps. At each step the risk can be minimised, and any faults contained within a small part of the total system.

4 Architecture

The overall model for Financial Services includes two major components characterised by the environments in which they run, as shown in figure 4.

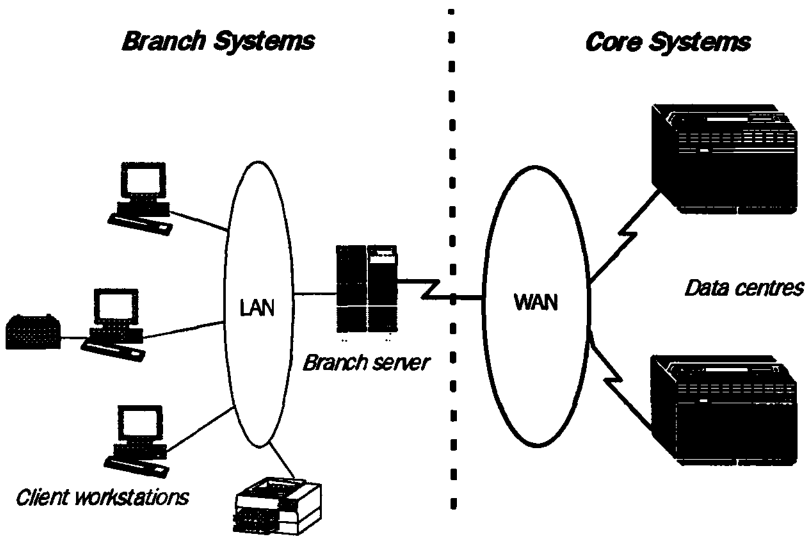


Figure 4 Overall Architectural Model

Most organisations find it impossible to change all elements of their existing systems at the same time. It was necessary, therefore, to devise an approach that would allow individual banks to choose their own priorities and timescales for what should be changed.

For most organisations there is a distinct break between those systems implemented at the centre and those implemented at the branch.

ICL's approach has been to develop an overall architecture for retail banking, based upon OPEN*framework*, encompassing both branch and core systems. This architecture takes client-server and object-orientation as two fundamental principles and comprises object models of both the retail banking business and the associated IT gramework required to support the business functions.

This approach facilitates not only the development of systems which can be deployed in a variety of environments but also ensures it is possible to develop and market core and branch systems independently whilst having the confidence that they form part of one consistent and compatible whole.

The next two sections describe two scenarios for implementation.

5 Core Systems Scenario

5.1 Introduction

The core systems scenario covers the development of the central systems. It will normally be a requirement that this can be done without changing elements of the branch systems and network.

The core systems in retail banking organisations have traditionally run on mainframe systems located at a small number of data centres. On-line transaction processing (TP) systems are used extensively by such organisations to provide strategic, mission-critical applications. They allow large numbers of users to access and process the organisation's data, provide predictable response times and offer high levels of application availability and integrity and security of data.

Access to these systems is required primarily from within bank branches and is traditionally provided by clusters of terminals accessing a central mainframe.

Many banks are considering a migration away from this mainframe centred approach for a number of reasons:

- applications have become tied to out-of-date technology
- the raw price/performance ratio of mainframes has not kept pace with that of PC and mid-range systems
- core applications cannot match the required rate of change in the banking business and have become obstacles to progress.

This last point is probably the most crucial. Applications have been in use for many years, and have become increasingly convoluted and fragile as statutory changes have been applied and new banking services have been added.

5.2 Approach

The client-server approach to design and implementation can increase the potential for change in core banking systems. Figure 5 illustrates ICL's design for core banking systems, known as ORB*it*. The central TP environment is itself divided into client and server components. These are co-ordinated by the distributed application services provided by TUXEDO®.

Although TUXEDO has been chosen for initial implementation, the design model makes no assumptions about the specific technology used to implement the Distributed Application Services (DAS) layer.

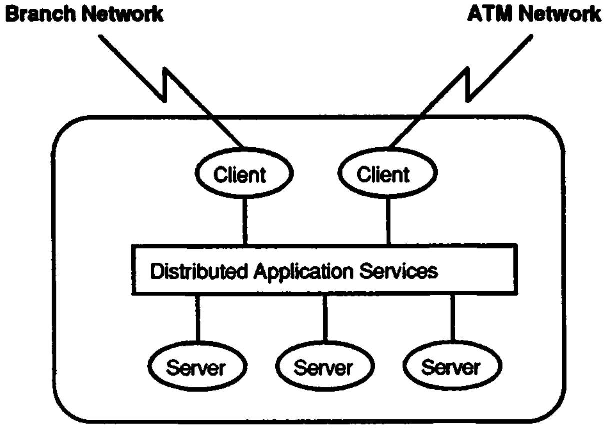


Figure 5 Model for Core Banking Systems

If other DAS mechanisms (for example, object request brokers, DAIS™, DCE) become appropriate, they can be adopted without change to the application code.

In this model, clients receive transactions from the bank's sub-systems, (for example, the branch network, ATM network, etc.), transform them into server requests, route them to the appropriate servers, collect responses from those servers and send a reply to the originators of the transactions. The servers are responsible for processing the requests they receive, including any database updates required

The branch subsystem could be implemented using a variety of techniques including one based on the client-server approach that is outlined in the branch systems scenario described later.

This approach enables the bank to move the central TP environment to an open systems platform without any modification to existing branch systems. It is possible, of course, to move these client processes out into the branches and generate service requests directly without intermediate transactions.

5.3 Distributed Application Services

TUXEDO's distributed application services provide the infrastructure within which this client-server environment can be constructed. They allow clients to issue service requests, route each request to an appropriate server and relay its reply back to the originating client. It also supplies many of the other attributes necessary for an open systems commercial on-line transaction processing environment.

Location Transparency	Service requests are routed by the DAS element, removing the necessity for the clients to be aware of where the servers actually are. This enables server reconfiguration without any disturbance to the clients.
Replication Transparency	The clients do not know how many instances of a particular server are running. The servers are capable of receiving requests from more than one client. Individual servers may be replicated to provide increased system throughput and high resilience configurations with service requests being processed by two or more servers.
Access Transparency	Clients and servers may be running on different platforms which impose different rules regarding data representations. This transition from one environment to another takes place in a way which is transparent to the components concerned.
Distributed TP	TUXEDO's management of global transactions ensures that transactions using more than one service, possibly running on different machines, are either all committed or all rolled back, thus ensuring data integrity.
Load Balancing	Where there are multiple, functionally identical servers running, the workload is spread evenly among them.
Systems Management	Facilities are provided to manage the operational environment. Additional servers can be started automatically if workloads increase sufficiently for service response to deteriorate beyond a preset threshold. Servers can be restarted automatically should they fail. Servers can be moved from one machine to another without affecting the clients.

5.4 Preserving The Strengths Of The Traditional TP Environment

High System Throughput

The transaction processing engine must service the high volumes of transactions which retail banks process. One way of ensuring this is to exploit replication transparency and load balancing by spreading the workload between many mid-range servers. Another is to use large processing engines, such as ICL's *GOLDRUSH* massive parallel processor. A combination of the two approaches could also be used.

High Availability

Retail banks demand greater availability from their core banking engines, especially as banking services are increasingly used 24 hours a day. This requires the elimination of the traditional batch processing periods by interleaving on-line transactions with batch activities. It must also be possible to back up databases while they remain available for transaction processing and to perform systems maintenance while continuing to provide on-line services.

Replication transparency provides facilities for transactions to be processed by more than one server. If these servers are placed on separate machines and one should fail then the transaction can still be processed by the other, thus preserving system availability. The requests received during this time are stored and forwarded to the other server once it is recovered.

When a machine needs maintenance, the servers located on it can be moved onto another machine.

5.5 Migrating to a Client-server Environment

To benefit from a client-server environment, an organisation must move its workload from the current system to the new; this is a non-trivial task. Any architecture must provide a clear migration path.

Often, the fragility of existing applications will preclude changing them just to support migration to a new environment. With TUXEDO, gateways can be created to these systems enabling transactions to be routed either to current systems or to new TUXEDO server applications. By using parallel running, transactions can be routed to both old and new systems. The routing may be altered by changing TUXEDO's configuration tables rather than by changing any application code.

However, migrating old applications to the architecture defined above may not be the first step. Perhaps a new application will be introduced using the new architecture, with TUXEDO enabling coexistence between the old and the new. Typically, a bank's existing systems will be account-based, with customer data being replicated across each system for which the customer has an account. In this environment, changes to customer data do not keep in step across the applications and the bank will be unable to obtain a coherent view of their customer base.

This is leading some institutions to build a customer database, linked to existing applications. This database is intended to support sales and marketing activities. Different client applications may need to access this database to reflect the way that different business processes are mapped onto the organisation. For instance, marketing and mailshot operations might be a central function, some account-based sales may be from branches and other more complex sales of financial services may be made by mobile representatives working from regional centres. There may be a telephone support centre where service staff may need to retrieve details of the latest customer contact from any of these sources. The ORB*it* client-server architecture is ideal for supporting these different application styles whilst normalising and rationalising the data.

5.6 Some Typical Configurations

The ORB*it* client-server architecture provides a number of different configuration options to support whatever the current business needs may be.

One is for the core system to remain centralised in one location, either on a single large open systems platform, a massively parallel open systems platform or a number of inter-connected smaller servers. This is shown in figure 6.

An alternative is to distribute the servers across a wide area network either at regional or branch level, as shown in figure 7.

The architecture permits gradual migration of the workload from the centre, to regions or ultimately, if required, to branches. A typical configuration would encompass more than one of the above options.

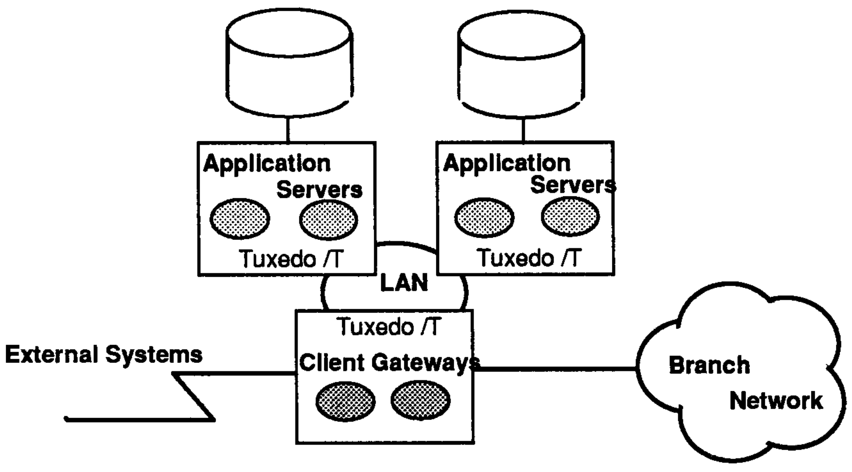


Figure 6 Centralised Approach

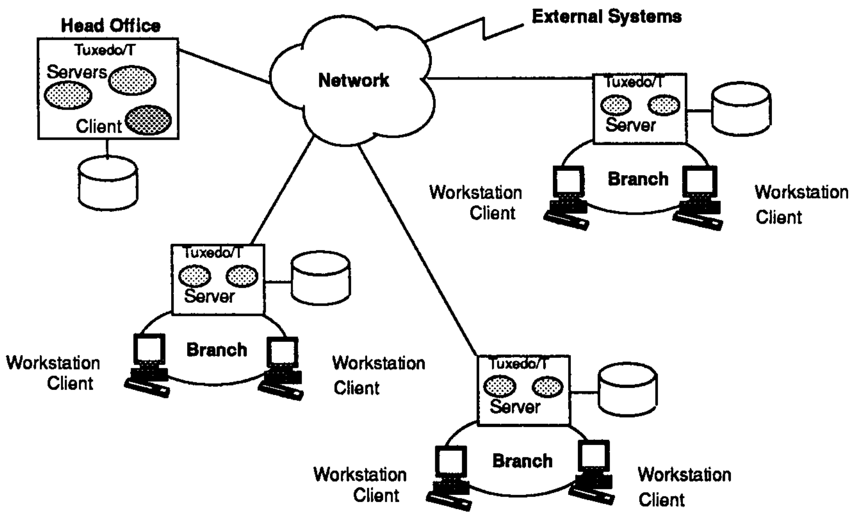


Figure 7 Distributed Approach

6 Branch Systems Scenario

6.1 Introduction

The branch systems in retail banking organisations have, in the main, been either completely free-standing, or simple terminals used to access the core mainframe systems of the type mentioned earlier in this article.

There is increasing demand for the branch-based systems to be far more flexible and user-friendly. This arises from the need to obtain both higher levels of service to customers and increased productivity for bank staff.

This has resulted in demands for new methods of accessing bank systems. For example, customers are making greater use of terminals in branches, and are increasingly likely to conduct their banking from home or the office via telephone; travelling sales people are using laptop computers. Ideally it should be possible to develop widely varied methods of access without having to make fundamental changes to the core systems.

Furthermore, it is also desirable, in the interests of cost and efficiency, that these different methods of access should be constructed, as far as possible, from the same, reusable components. Developments in client-server and object-oriented software technology, together with advances in PC hardware technology, mean that it is now possible to deliver such systems in a cost-effective way.

6.2 Current Situation

The technology used within the branches of a typical western bank has normally evolved over a number of years. The IT for a typical branch would look something like that shown earlier in figure 2.

This relatively simple diagram conceals considerable diversity and complexity, in both hardware and software terms. Over the past 30 years, successive layers of technology have been added on top of existing systems until, in the words of one bank's IT director, *the IT system resembles an archaeological dig!*

Even when systems are physically co-resident they are typically not integrated with each other in any way. The ATM network for example is often a free-standing network that does not use any of the IT infrastructure that has been installed for the branch. At best it may share some of the communications infrastructure.

6.3 OMNIA Client-Server Model for Branch Systems

The branch part of the ICL Financial Services solution for retail banking is called OMNIA™. The model for OMNIA divides the branch based system into a further two tiers, as shown earlier in figure 4.

In this model, the branch software can be considered to be split into two distinct classes, those resident on the client and those resident on the branch server. The client and the branch server are always located in the branch. Access to the core systems across a wide area network is always via the branch server. The core systems are then treated as another layer of server and are accessed in a consistent way from the viewpoint of branch applications.

The core systems may be existing systems based on traditional mainframe technology. These are accessed via a software interface which hides the details of the core systems implementation. This allows them to be accessed in a client-server mode. Alternatively, the core systems can be more modern systems that use client-server architecture as outlined in the previous section. In this case the main difference is that there will already be client-server interfaces in place, and it will not, therefore, be necessary to develop special code within the branch systems to present a client-server interface to the core systems.

Key to this model is the fact that applications resident at the branch level should be largely independent of whether the core systems have been implemented using traditional mainframe technology and techniques, or whether they are based on the client-server and distributed processing techniques.

Client systems should be based on common technology to provide benefits in terms of integration, economies of scale and ease of implementation and support.

The current variety of systems resident in the branch are all refocussed onto a single applications infrastructure. This is flexible enough to support the wide variety of applications and user interfaces that current and future systems will demand.

6.4 Branch Servers

The branch server provides those elements of functionality that are either common to all workstations in a branch, or which it would be uneconomic to provide at each workstation.

Another key function of the branch server is to provide a client-server method of interacting with the core systems. These core systems are isolated behind a software interface. This allows applications to be developed within the branch, independently of the method of communicating with the core systems.

6.5 Client Systems

A central part of client systems is the concept of a *workplace*, i.e. an environment designed to support a specific job role within the organisation such as a teller or a sales advisor, or even a customer using banking services via an ATM or over the telephone.

It is important to understand that these terms are not hard and fast definitions. In practice, each organisation will define the roles, and the processes which they perform, to suit their business needs. This means that ICL must be capable of delivering extremely flexible and configurable systems.

Figure 8 shows a conceptual view of a workplace.

A workplace comprises both hardware and software with, as far as possible, much of the generic functionality supplied by common components, such as user interface enablers, access and authentication facilities and workflow software. Office functionality, such as electronic mail and word processing, may also be shared between workplaces.

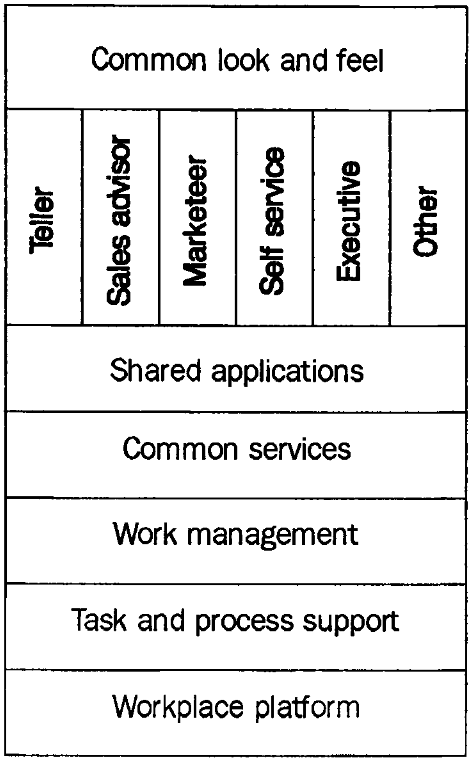


Figure 8 Conceptual View of Workplace

OMNIA makes use of standard ICL and third party products wherever possible so that ICL Financial Services can concentrate on the provision of those added value components which are either specific to the financial services industry, or are required to provide the flexibility and potential for change that retail banks demand.

It is also a goal that the same business functions can be delivered through different sales channels using the same software technology. For example, the software for withdrawing cash from an ATM should, except for obvious exceptions due to the nature of the environment, be the same as that used to implement a manned teller workstation or a telephone banking system.

This strategy leads naturally to the adoption of object-oriented methods of design and implementation.

Central to this strategy is what we have called the *OMNIA Banking Frame*, which comprises the business and system objects and services, together with the tools required to enable application developers to create business solutions to meet the requirements of each individual organisation.

OMNIA Banking Frame

The OMNIA Banking Frame comprises

- *a retail banking business model* which defines the business objects. (Note this covers both branch and core systems)
- *a set of business objects*, which support specific business functionality relevant to financial services, such as cheque and till managers, as well as generic objects such as customer
- *an IT framework object model*, which defines the service objects and the environment in which the business objects operate. (Note this covers both branch and core systems.)
- *a set of service objects*, whose role is to insulate the applications from the environment, thereby maximising the potential for change. Examples of service objects are Database Manager, Screen Manager, Security Manager, Host Manager
- *a set of development tools and design templates*, to enable users to exploit the business and service objects to create a wide range of financial services applications. This development environment covers all the activities from business process definition through to actual implementation of software systems.

The OMNIA Banking Frame has been designed to provide maximum flexibility for organisations in the way in which they implement their retail banking branch systems. While the business objects themselves may be configured to provide a range of business functionality, the service objects within the IT framework model insulate these business functions from the environment, such as database access and distributed application services.

At the branch level, database access is required for journalling purposes at branch server level, and is currently implemented via a service object. However, in the future we anticipate that Microsoft® Open Database Connectivity (ODBC) will be the preferred solution.

Distributed application services are currently provided either by SQL (for client-branch server interaction as described above) or by Advanced Program to Program Communications (APPC) protocols for interfacing with core systems. This allows the client to emulate existing interfaces with hosts without the need for any changes to existing mainframe applications. For environments with UNIX servers, TUXEDO provides the carrier capability; in the medium term we anticipate using object request broker technology to support access to heterogeneous servers (see previous section for more details on server strategy).

6.6 Migration from Existing Systems

Most organisations are not in a position where they can make a completely fresh start. A key requirement is that there is a clear and sensible way forward from the existing systems to those based in the client-server techniques identified in this paper. This may be achieved in a number of ways:

- PC based systems can be used to emulate existing methods of working. By choosing the appropriate mix of hardware and software it is nearly always possible to emulate the existing system with the minimum of change. The PC may simply emulate a terminal connected to a mainframe, or it may be required to perform more complex tasks.
- Additional functionality can often be obtained from the emulation of existing systems simply because they run within the same operating environment and infrastructure as the new client-server applications. By using encapsulation techniques, the old and new applications can interact with each other.
- New systems are then developed which operate naturally in client-server mode. These co-exist with the older systems which will gradually be superseded by new ones of higher functionality that have been developed using the client-server approach, and object oriented design.

Furthermore, this approach applies to migration of both business functionality and technology. As described earlier, business functionality is clearly separated from the technology which supports it enabling technology migration and evolution to take place alongside the evolution of the business functions.

7 Conclusions

ICL Financial Services Systems intends to be at the forefront of those who are providing systems to the retail banking marketplace. The strategy is to exploit client-server technology and associated techniques, such as object-orientation, to build flexible cost-effective solutions that can be adapted to the rapidly changing business and technology environment.

The client-server approach to building systems is flexible enough to be applicable to addressing the needs of the central core systems and the geographically distributed branch systems. This is a very powerful endorsement of the basic paradigm behind client-server computing.

References

ARCHER, B. The Management of Client-server Systems, *ICL Tech J,* Vol. 9 Iss. 1 pp. 122-137, 1994.

BRENNER, J.B. Client-server architecture, *ICL Tech. J.* Vol. 9 Iss. 1 pp. 3-17, 1994.

DAY, R.P. How ICL Corporate Systems support Client-server: an Architectural Overview, *ICL Tech. J.* Vol. 9 Iss. 1 pp. 18-46, 1994.

Definitions

ORBA - Open Retail Banking Architecture, ICL's overall business and technical architecture for Retail Banking

OPEN*framework* is ICL's systems integration architecture and methodology.

Acknowledgements

TUXEDO is a UNIX transaction processing system available on a wide range of platforms and is capable of interworking with PC client systems and a range of servers. TUXEDO is a registered trademark of UNIX Systems Laboratories, Inc., in the U.S.A. and other countries.

DAIS is an application development and systems integration tool kit for building client-server applications in a multi-vendor environment. At its core is an object request broker. DAIS and OMNIA are trademarks of International Computers Limited.

DCE is the Distributed Computing Environment from the Open Software Foundation

Biographies

Mike Haynes

Mike Haynes joined ICL from university in the late 1960s. Apart from three years spent with a micro computer manufacturer he has worked for ICL ever since. His career has been spent in various software roles, including compiler writing for 1900 and early 2900 Series computers, writing software for ICL's Distributed Array Processor and 10 years spent managing all aspects of ICL's mainframe and mid-range CASE (application development) programmes.

Since the beginning of 1993 he has been responsible for the development of the systems architecture for ICL Financial Services Systems.

Geoff Ibbett

Geoff Ibbett joined ICL as a graduate trainee in 1984. His career has been primarily in software development and product consultancy, with a break of two and a half years working outside ICL. Since joining ICL Financial Services Systems in 1991, he has held positions as a banking product consultant, product integration manager and latterly as designer of a core banking system based on client-server approach, online transaction processing technology and object oriented techniques.

Dave Walker

Dave Walker joined ICL from university in the early 1970s. He has worked throughout this period in the various ICL units that have been responsible for selling systems in the financial marketplace. He has always specialised in systems that involved significant amounts of networking or integration expertise.

His current role is as a Systems Integration Consultant to ICL Financial Services Systems with particular responsibility for technical issues involved in implementing client-server systems for customers.

A Practical Example of Client-server Integration

Andy Ellis
OPEN*framework* Division, ICL, Bracknell, UK

Abstract

Systems Integration is often viewed as primarily a technical problem, yet many of the issues may not be technical. In this paper a case study is presented which describes how a problem of integrating an existing system into a new client-server environment was resolved. The problem had been perceived as technically difficult yet was resolved by analysing the business environments within which the systems operated before designing a technical solution.

1 Introduction

The spread of client-server architectures is bringing an increased degree of information systems integration to many organisations. The effects of a move to a client-server architecture are relatively easy to manage when changes are localised to a single department. However, when the changes extend across an organisation many more factors come into play, many of which were seen by the manufacturing industries in the early 1980s when they moved to Computer Integrated Manufacturing.

Isolated *islands of automation* gave way to integrated *design to delivery* processes with an integrated business system managing the automatic transfer of information between the different sub-processes. These changes were driven by a need to reduce costs to cope with tough competition world-wide. There was also a rapid rate of change in demand, in technology, and in customer expectations [Ranky, 1986]. Finding a solution quickly enough became as much a question of business survival as of competitive advantage. Service industries now face similar pressures.

When organisations move from disparate *islands of IT* to some form of distributed client-server architecture, there may be business and political impacts as well as technical. This paper seeks to explore some of these impacts by examining a recent case study in which the need to integrate disparate systems arose from a merger of two businesses, one of which was changing to a distributed client-server architecture.

2 The Case Study

2.1 Initial Assessment

The management of two merged businesses believed that it was inefficient and uneconomic to run two equivalent, but different, IT systems side by side to support a combined business operation and decided that the disparate IT systems should be integrated. They saw this as a technical issue and so requested a systems integration plan that would move them to one organisation supported by a single seamless IT system. The questions were - what choices should be made, when, and why?

Company A was a roadside vehicle repair and recovery services organisation with a number of regional Response Centres taking telephone calls from customers and deploying the required assistance. Each Response Centre was taking some 15,000 calls per week on average, but with substantial variation in load according to time of day and weather conditions. In the past, the Response Centres had used dumb terminals connected to small local mainframes, linked in turn to the corporate administrative mainframes (figure 1), but software changes had been taking too long to implement prompting a move to the greater flexibility and capability for rapid change generally associated with a client-server architecture.

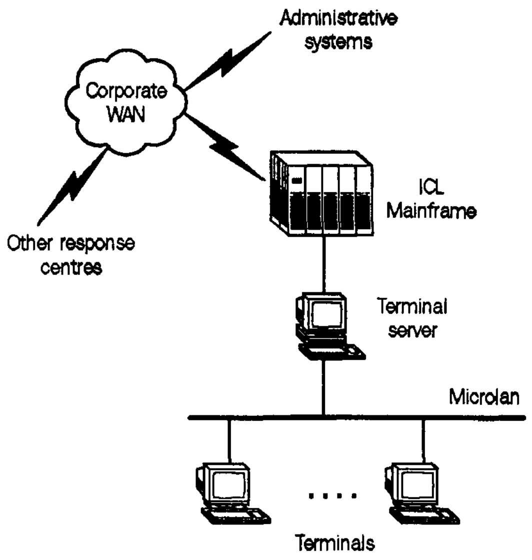


Figure 1 ICL Mainframe response centre environment

The new distributed client-server system was based on UNIX servers and PC workstations (figure 2), and was being implemented in-house. This used a Windows-based call-handling front-end on an Intel486 PC using services on local UNIX servers but with direct access across the corporate network to servers located at other Response Centres to provide resilience. Communication with the administrative systems still running on corporate mainframes was via terminal emulation protocol (TEP) services supported on these local UNIX servers for interactive queries, and via a modified TP service for automatic transfers.

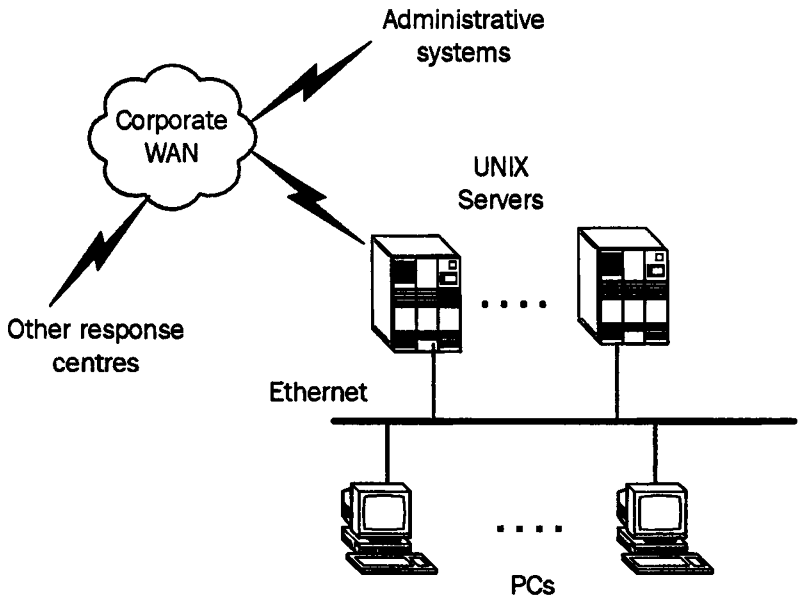


Figure 2 Client-server response centre environment

Company A was profitable and had decided to expand its business by acquiring Company B.

Company B specialised in deploying roadside repair and recovery services for commercial vehicles, up to and including heavy goods vehicles, and had valuable experience of using IT effectively in this type of business. However, it was a much smaller company and had just one Response Centre taking some 2,000 calls per week on average. It was successful in its chosen market segment using an IT system developed in-house and running on an IBM AS/400. The IT system was supported, maintained and enhanced by a small and dedicated development team, proud of their

IBM skills and of the success of 'their' system in supporting the business. Intel486 PCs were connected to an AS/400 connected in turn to an administrative system, again an AS/400 (figure 3).

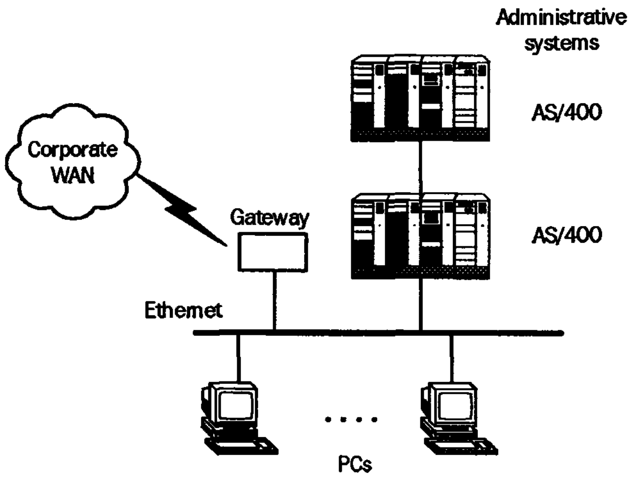


Figure 3 AS/400 Response centre environment

To reduce costs and improve efficiency, the management of the combined company decided that any customer call would be taken at any Response Centre and, if necessary, transferred to another, without being constrained by the technology. It was accepted that this would probably be limited in practice so that calls from a particular region would generally only be redirected to centres in adjacent regions. This was part of an objective of one business supported by a single, seamless information system. In particular the Response Centre from Company B would be amalgamated physically with one of the Response Centres from Company A. Many of its enquiries were then to be directed to Company A's other Response Centres.

The management were now looking for a technical integration plan which would lead towards this objective.

2.2 Evaluation using OPEN*framework*

The first stage of the study was to gain a broad understanding of the nature of the IT systems from each company, what facilities were provided in the Response Centres, and how much discrepancy there was between them. It was important to understand the current situation in sufficient detail. It was not practical to discover everything that might have an impact, but it was essential to minimise the risk of major factors being overlooked.

The approach was based on analytical models from OPEN*framework*, and on the method outlined in figure 4. The scope of the study was restricted to the Response Centre operations, excluding the administrative systems. The main force for change at this time was the desire to amalgamate the Response Centre from Company B with one from Company A.

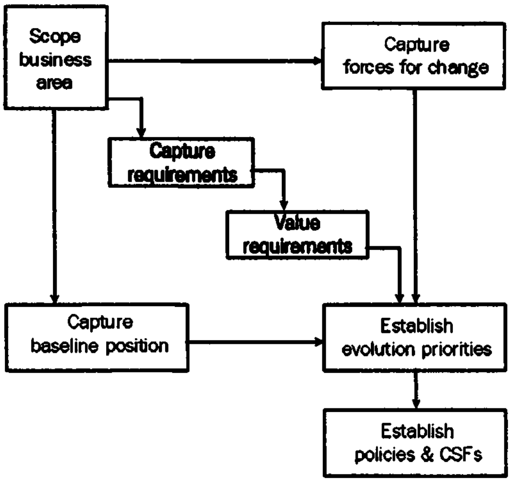

Scope business area
Capture forces for change
Capture requirements
Value requirements
Capture baseline position
Establish evolution priorities
Establish policies & CSFs

Figure 4 Evolution Planning method

The basic situation as described appeared to offer an immediate answer to technical integration of the IT systems. Using the new distributed client server system it appeared likely that, with a suitable choice of terminal emulator on each PC and a gateway to provide the necessary translation, the AS/400 could become another server accessed across the network from each PC as required and fitting naturally into the client-server model (figure 5). Further AS/400s could be added at other Response Centres to

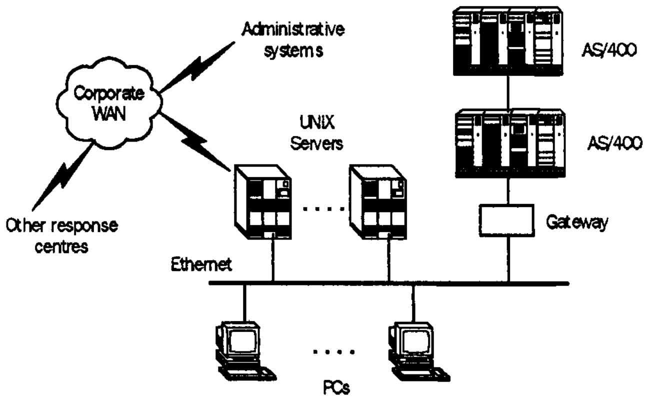


Figure 5 Integrated environment

provide additional resilience and to cope with anticipated business expansion. This would be a relatively straightforward answer to the immediate need for operational integration of the Response Centres although not fully answering the requirement for seamlessness because of functional difference between the two company systems.

2.3 Analytical Approach

This approach made it possible to assess whether seamlessness was important as well as identifying other requirements. Individuals from each company were identified representing the business and technical perspectives from OPEN*framework* - Enterprise Manager, Customer, Trading Partner, Employee, User, Application Developer and Service Provider. Due to the sensitivity of the study, the customer perspectives were represented by individuals from within the company; this restriction was judged to be acceptable in this case. Requirements capture from the other perspectives was carried out through individual interviews due to the limited time available and the limited availability of the people representing each perspective. Normally this phase would have been carried out through workshops to reduce the time taken to resolve disputes and conflicts.

The business purpose to be achieved by integrating the systems, the benefits expected and the business related capabilities that should not be lost were captured, with the terms of reference for the study providing the context. Integration priorities were expressed in terms of the OPEN*framework* Qualities. As a control, the critical success factors, risks, and mission-critical factors for each perspective were also ascertained.

In addition, each interview identified extra people whose views should be considered. In the early interviews a number of further people were identified, but few new names appeared from the later interviews. This approach provided a check on completeness. In the event, time proved too short to interview everyone but the OPEN*framework* model of a limited but balanced set of perspectives gave confidence that a sufficient subset of individuals had been questioned. Few interviewees could cover all areas, but the extent to which information was duplicated by the later interviewees gave confidence that all relevant areas had been covered and in adequate detail.

2.4 Analysis of the Information

The high volume of information collected, some 15,000 words of abbreviated notes, was then reviewed and assessed for completeness. At this stage, and with this volume of information, it was difficult to be sure where the gaps were. To digest this information it was converted into meaningful one line statements, each holding a single message, such as **Management should be able to divert calls to alternative response centres for managing workload.** This reduced the information to some 450 one line statements. By partitioning them under the headings - Vision, Forces for Change, Risks, Benefits, Critical Success Factors, Existing Plans, Requirements, and Solutions, they were rationalised to approximately 350 one line statements and 30 suggestions towards solutions.

The many requirements were further divided across the OPEN*framework* qualities and technical elements. The requirements at a business level were divided according to the model in figure 6. This model was derived by relating the technical elements of OPEN*framework* to the five layer model of business transformation [Kay, 1993], Deming's work on statistical process control [Neave, 1990], and recent changes observed in business practice [Peters, 1988 and 1992]. Finally, each one line statement was checked to ensure that it lay within the scope of the study before being linked with the IT environment more affected by that particular issue. As the picture built up some clear messages became apparent.

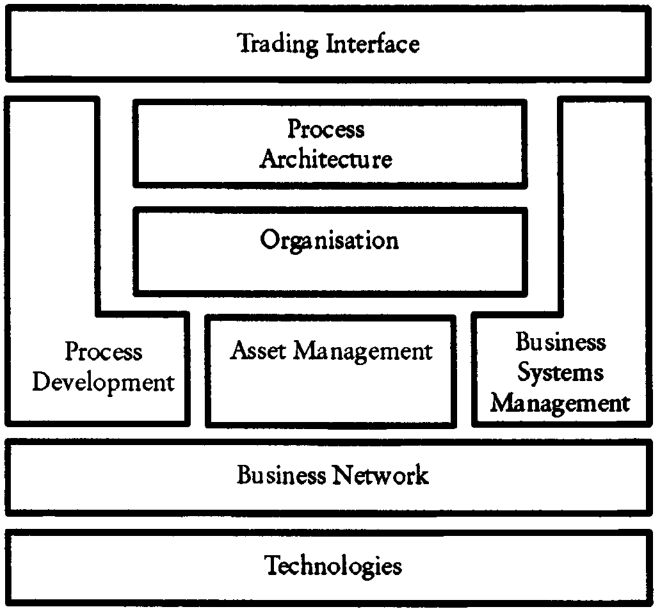


Figure 6 Business requirements model

2.5 Other Issues

The analysis showed that further technical detail was needed from the development teams and from suppliers of specific products to uncover technology opportunities and to identify likely technology constraints. It is not unusual to find an apparently minor technical detail which is a major hindrance to an otherwise appealing technical solution. By interviewing relevant technicians it was discovered that there was a subtle difference in the use of the ethernet protocols in the two environments. This difference was likely to cause difficulties with routing messages across the wide area network. As a result, extra constraints were added to the Networking Services section of the analysis to ensure that this problem was avoided.

The approach also made it straightforward to eliminate duplication. Any conflicts identified were followed up, either with the individuals who originated the conflicting requirements, or through relevant working parties and committees.

By this stage the broad picture was well understood which made it much easier to focus on the relevant details from the mass of hardware and software configuration information available. The high level verbal descriptions of the current operational systems were expanded by obtaining detailed configuration diagrams which identified the specific combinations of hardware and software components being used. It became clear that the new client-server system of Company A was still under development and had not yet been installed in any of the response centres, although the first live test environment was about to be implemented.

It was now possible to understand more fully the implications of the planned migration path from the existing mainframe-based environments which were still active to the new client-server system. In particular, the stable and effective systems from company B were expected to undergo substantial change to be integrated with a client-server environment that had not, at that time, even been tested in a live environment, and parts of which were still very much under development. A further complication involved the interfaces between the administrative systems of the two companies. These were outside the immediate scope of the study, yet had to be included in the integration plan, at least in outline. Without them it was impossible to appreciate in full the nature of Company B's operations. On-line access to their administrative system was necessary for deciding whether a particular enquiry would be accepted.

3 A Fresh Perception of the Problem

3.1 Management Issues

It was now clear that the situation was far more complex than had been described at the start of the project. As the management's intentions for the integrated system were not consistently understood, each person involved was interpreting their part of the problem in a different way. No one person had a clear overall view of all the relevant factors at business, political and technical levels. Yet the analysis had shown that the operations from Company A were likely to gain significant business advantage with minimal risk, whereas the operations originating in Company B faced a major upheaval which was to bring them almost no benefit and could diminish their competitiveness. There was, therefore, much resistance to the proposed changes from Company B's staff and it was this that had been blocking progress.

The two businesses were in many ways dissimilar, and what had appeared to be almost identical business processes were actually very different. On the one hand (Company A) the emphasis was on handling a high volume of calls as quickly and economically as possible and at minimum cost, typically taking just 2 minutes per call. On the other hand (Company B) the emphasis was on providing a high quality service with personal attention. In some cases it was possible for a single call to incur the best part of a day in finding a specialist supplier with the right skills and spares.

So the operators in the Response Centres from Company A were essentially production line workers handling calls as quickly as possible whereas in the Response Centre from Company B the operators were highly skilled specialists dealing with calls from professional drivers about a wide range of commercial vehicles. The two businesses also differed dramatically in the time scale of change in their markets, 6 months as against a few weeks; the commercial vehicle market was very competitive and undergoing rapid change, whereas the personal market was driven more by broad market trends.

These differences raised a number of issues that had to be resolved before an appropriate technical solution could be designed. One of the highest priorities was to decide how the operational integration of the centres was to be managed and what form the merged Response Centre operations should take. Because of the operational differences between the two types of response centre it could not be a straight amalgamation; the nature of the operational tasks meant that operators would have had to switch continually between very different types of work, degrading their quality of response for high-value calls whilst adversely affecting their productivity on high-volume calls.

After discussion and negotiation, particularly with staff familiar with both Response Centre operations, it was found possible to combine the activities of the two types of Response Centre in a way that provided management with sufficient operational flexibility. In essence this consisted of having a pool of production line operators together with a section of more specialist operators. This gave the operators an enhanced career structure. With some other changes at supervisory and management level as well, all levels of staff were encouraged to make the change succeed.

A less immediate priority was to reduce the variety of types of hardware platform where possible. Senior management expected the technical choices to lead strategically in this direction although pragmatic technical anomalies were acceptable if they were sufficiently low cost and did not obstruct the longer term view. This made it acceptable to consider a temporary combination of terminal emulators and communications software stacks on a limited number of PCs to offer flexibility in the changeover period. This appeared likely to give early operational benefits as the chosen UNIX platform could be introduced as a gateway between the two environments to provide remote access to the AS/400 from the client systems.

3.2 Application Development Issues

The two businesses also used different application development methods, tools and management processes. Company A had a sizeable development team using standard large team management and development methods supporting a phased client-server implementation, optimised for handling high throughput volumes. Changes to the applications were made on a six-monthly cycle. New requirements were derived by another department from statistical analysis of market trends. In contrast, Company B had minimal and lean management processes and used an integrated CASE environment to support a development approach of rapid prototyping followed by swift asynchronous implementation and roll-out within weeks of customer-demanded changes. Prototyping was usually carried out in the presence of someone from the particular customer business requiring the change.

Thus, as well as the operational issues within the Response Centres and technical issues about integrating dissimilar IT environments, there were now human factors and management issues from the development environment. To insist on applying in full the more bureaucratic management methods of the large development team to the working environment of the small, rapid response team would have destroyed one of that team's key strengths - flexibility. A fast-track simplification of the management processes was needed that would provide sufficient control with fewer change reviews by management. The development environment needed to allow the use of both rapid prototyping and phased delivery techniques in parallel. A suitable organisational structure would encourage differences between the two teams rather than leading to uniformity and would be necessary for success in the two very different markets. The organisational structure recommended is shown in figure 7.

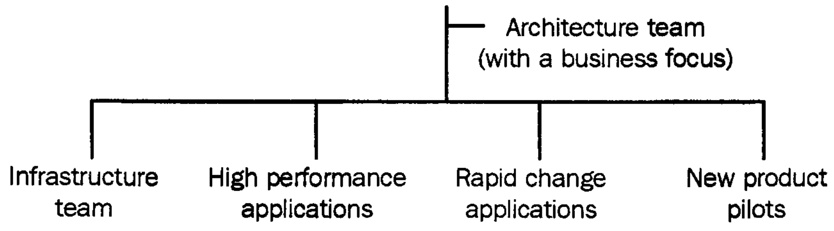


Figure 7 Proposed organisation

3.3 Organisational Structure
The purpose of the architecture group was to ensure that the systems would continue to support the business throughout the changes. To achieve this it was recommended that members be drawn from the business units as well as the information systems departments. Their skills could be enhanced by appropriate education and training. The infrastructure team would specify and police the middleware, the infrastructure and communications components needed to enable applications on a particular workstation to use services from a number of servers. They would ensure that standardised services to support the various distributed applications would remain independent of the short term needs of particular applications. This would help in maintaining the overall system's potential for subsequent change by defining *firewalls* between parts of applications, thereby limiting the impact and scope of subsequent change. The three remaining teams focused on developing pilot applications for new product opportunities, on developing readily changed applications, and on producing optimised high volume applications.

The clarity that resulted from this analysis provided a common architectural view. The associated suggestions for technical activities encouraged the two technical teams to work together to a common plan and to take on joint responsibility for the ongoing technical aspects of integration.

4 Conclusions from the Case Study

4.1 Increased Complexity
The transition to a distributed client-server architecture brought to the fore many of the issues and advantages that arise with Open Systems. The islands of IT previously supporting the individual business functions were now being brought together in enterprise-wide information systems to support a rationalised business operation. The changes to the business accompanying the integration of these disparate IT systems, possibly including subsequent dispersion into separate operating companies, brought additional levels of complexity to the whole process of change. Business, political and technical factors all had to be considered for success to be achieved.

With so many factors being relevant, it was necessary to adopt an approach that had a sufficiently wide scope. In this case OPEN*framework* provided the necessary breadth and helped to partition the problem into manageable sections; it also provided a structure for managing the high volume of information involved. The completeness of its models helped to show up significant gaps in the requirements specified allowing an opportunity for these omissions to be rectified. These gaps, including some which, when filled, changed the requirements on the middleware infrastructure, would have been extremely costly to correct if they had been left to a later stage in the integration project. Most of them arose from the wider business context and so would have been missed if a purely technical approach to systems integration had been taken.

4.2 Change in Business Processes

Full benefit was unlikely to be achieved merely by integrating the underlying information systems. The client-server approach encouraged much greater movement of information around the organisation, together with a wider spread of the functions provided. This introduced opportunities to combine, modify and improve the operational processes within the business. As a result, political territories within the business were likely to alter as roles and responsibilities changed.

4.3 Middleware is Significant

With more technology suppliers adopting an Open Systems approach technical issues were becoming a smaller proportion of the integration problem. However, there were still significant middleware conflicts where so many factors met. This was an area where clear firewalls had to be established and maintained, so needed to be highly valued by management. This was to be achieved by setting up a specific middleware team.

4.4 Coexistence

The last conclusion reached was that, as the existing systems would have to co-exist with the replacement systems for some time, there would be a period during which extra operational overheads would outweigh the anticipated returns. As the full business benefits would only be achieved once the existing systems began to be decommissioned, management would have to maintain strong commitment throughout to capture the potentially rich rewards.

5 Recommendations for Other Projects

The conclusions from this study indicate areas which other projects might benefit from considering. There are some further general recommendations which can perhaps be drawn from this study.

5.1 Business Policies

The many contrasts between the two businesses exposed a number of areas which needed to be addressed by company-wide policies. Policies were needed to ensure a consistency of approach throughout the period of change, particularly in regard to changes to the operator roles and career structure, and to clarify the way in which these roles would be supported by the integrated information systems. Although these policies were not directly aimed at the information systems, they were needed to ensure successful use.

5.2 Infrastructure Policies

The contrasts between the development teams and their development tools and methods also indicated a need for company-wide policies. The most immediate of these arose with respect to the middleware components. As the modification time scales required by one of the markets were so short, it was critical that swift and frequent changes could be made in the low volume applications. However, it was imperative that these changes would not have an undue adverse impact on the performance of the high throughput applications, even though both would be running across the same middleware.

This conflict should be managed from both business and technical levels to ensure that appropriate criteria are adopted in making trade-offs between the two. Benefits from investment in infrastructure can often be hard to quantify, yet it is fundamental to distributed client-server systems. It is recommended that this should be championed by a specific member of the architecture team who can, by understanding both business and technical issues in significant depth, develop and maintain a solid business case and ensure that the importance of the infrastructure is generally recognised and accepted.

5.3 Development Methods

One might assume that an information system will automatically be easy to change if it is based on a client-server architecture. Unfortunately, this is not the case. If the information system is seen as one development instead of as a series of co-operating components, the integrity of the firewall interfaces, which localise the effects of change and allow for heterogeneity within the system, may get degraded. Those firewalls should be specifically designed and maintained.

The architecture team, being responsible for the structure and integrity of the overall system, may be a natural and appropriate owner of this responsibility. With a broad overall view and an influential role, this team should be in a position to recommend those policies which will protect the firewalls and hence maintain the ongoing *potential for change* of the system. This approach also appears to support continued heterogeneity within the system.

5.4 Architecture

Finally, with *Openness* becoming more widely adopted in the IT industry, systems integration is increasingly about choice between viable alternatives and less about finding a single technical solution. Thus, one has to collect and understand sufficient information about the business context for cost and benefit comparisons to be carried out. This large volume of information can be managed successfully if the business and its supporting information systems are represented and documented in the form of a coherent model, an architecture.

In this particular study the company was able and keen to resume responsibility for the technical aspects of its integration problem as soon as it had the clarity of understanding that comes from such a representation.

References

RANKY, P.G., Computer Integrated Manufacturing, Prentice Hall International, 1986

NEAVE, H.R., The Deming Dimension, SPC Press Inc., 1990

PETERS, T., Thriving on Chaos, Macmillan, 1988

PETERS, T., Liberation Management, Macmillan, 1992

KAY, M.H., The Evolution of the *OPENframework* Systems Architecture, *ICL Tech. J.* Volume 8 Issue 3, May 1993

Biography

Andy Ellis

Andy Ellis is a Chartered Engineer. He graduated in Mathematics from the University of Cambridge and joined ICL in 1988 having worked in various capacities with large scale scientific systems for SERC and with telecommunication systems in STC. His work has covered a wide range from software and hardware development through to business consultancy. He is a Senior Management Consultant within OPEN*framework* Division.

From a Frog to a Handsome Prince: Enhancing existing character based mainframe applications

Alan Beer
Project Manager, Solutions Centre, OPEN*framework* Division, ICL, Bracknell, UK

Abstract

Many mainframe applications have been developed over the past few decades. Studies have shown that the life of an application and its associated data can be well over 20 years. Must we then retain dated character mode interfaces for the end user?

This paper outlines some of the practical possibilities for moving these existing character-based mainframe applications into a Client-Server mode of operation. In this mode one can then add Image; Black & White and Colour, Video; Pictures & Sound and many other technologies to enhance legacy applications with minimal, and in many cases no, change to the mainframe application itself.

This paper traces the experiences of one particular ICL Mainframe customer. They were interested in two aspects of the problem, how to develop such distributed Client-Server systems and what problems would be encountered. They chose to investigate, the move to Client-Server, by means of a workshop.

1 Introduction

A large number of organisations rely on operational Information Technology (IT) systems developed over 5 years ago. These systems were developed before the PC revolution which dramatically altered the PC price/performance and generally increased the capability of desktop workstations.

PC systems can now be bought at a very attractive price. It can be cost effective to replace some, or all, of the character only, green on black, dumb terminals currently in use. However one of the significant advantages of the mainframe-centred, dumb terminal, approach is that control is vested in one place in the organisation.

Where the user is known to have a PC the developer of an application can provide a more pleasing or friendly interface for a particular application or system. This interface allows the user to access the system using such facilities as graphics, windows, icons and a mouse. The user can make selections and perform computer tasks using a click of the mouse. Previously they may have had to type in textual commands to achieve the same effect.

Using this graphical mode the PC now has some intelligence running in it, in the form of a computer program. That intelligence is placed on the PC to make the user's job easier. But in placing programmed logic onto the PC we have introduced a level of complexity we did not have before. It also introduces a problem of coordination and control. How can we be sure that the version of the graphical interface is the correct one for this application? Who loaded it onto the PC? What else is on the PC? If something goes wrong, which part of the system is at fault?

Giving the user a PC both improves the situation and introduces potential problems. How then do you give the user a PC, in order to improve the interface, yet still retain control? To answer this question the paper traces the experiences of one particular ICL mainframe customer. They were interested in two aspects of the problem, how to develop such distributed Client-Server systems and what problems would be encountered.

After discussions with ICL, on the subject of Client-Server, the customer requested a workshop in order to build a prototype system. The idea of the prototype was merely to demonstrate the possibilities that such an approach might offer.

2 The Workshop background

The first question the customer asked was 'exactly what is Client-Server?'

2.1 What is Client-Server?

We all may believe we have a good idea of what Client-Server is and what it is not. Full explanations appear in other papers in this journal. A basic explanation is available in [Brenner, 1994].

The three terminals shown in Figure 1 are dumb, or legacy, terminals connected to a mainframe. All the intelligence is driven from the mainframe. A single terminal may be replaced by a PC. Is that Client-Server?

Using a Gartner Group definition, Ref [1], we see that Client-Server is the '*Splitting of an Application into Tasks that are performed on Separate Computers, one of which is a Programmable Workstation (e.g., a PC).*'

So separate tasks run on different (interlinked) computers with a specific task running on the computer for which it is best suited.

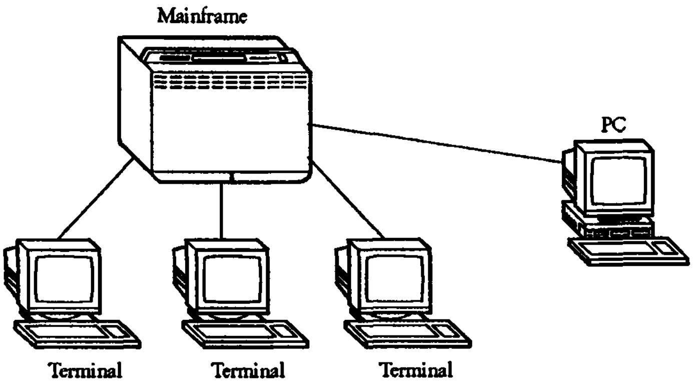


Figure 1 What is Client-Server?

We know that these tasks can loosely be grouped into Presentation, Application and Data management. Also we have come to accept that presentation is best run on an intelligent workstation or a PC with graphics capability.

So, to illustrate the definition of Client-Server, we could ask ourselves a series of questions, as follows:

Q. Is a situation where a (dumb) terminal accesses a mainframe application Client-Server?

A. No.

Q. Is a workstation running both an application and its graphics presentation but where the data is on a remote machine Client-Server?

A. Yes.

Q. What if we replace the terminal by a PC running Terminal Emulation (TE). Is that Client-Server? See Figure 1.

A. Our initial answer might be no, but let's look at the environment.

What if the TE supports word wrap? On a dumb terminal I may be typing in text. At the end of a line I may be surprised that a long word 'wraps round' to the next line. I may have typed 'another' but I may find that 'ano' is at the end of one line and 'ther' is at the start of the next line. On the PC, with word wrap, I would see that the whole of the word 'another' would be moved to the start of the second line with no truncation.

Similarly the PC may support basic local validation such as whether a field should be numeric or alphabetic etc. For example there might be the

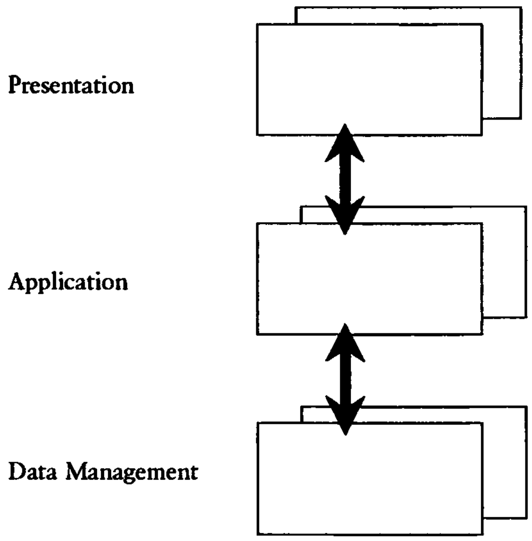


Figure 2 Separation into Presentation/Application/Data Management

requirement to input an international telephone number, which is numeric. On the PC if I type an alphabetic O instead of a zero then as I tab to the next input field I would see the telephone number highlighted to show that it was in error.

Performing such validation locally on the PC can often improve the speed and appearance of a character-based application to the user.

Using the Gartner definition, these and other facilities can be split out and run locally, on the PC. More complex cross field validation might check a Post Code against a county, an Invoice amount against a credit limit and so on.

The terminal emulator will almost certainly support colour, in which case one can re-map the single colour terminal screen presentation onto meaningful everyday colours. One might use blue for neutral background text, green for directional text, amber for warnings and red for serious errors.

Another advantage of a colour PC is that a partially colour blind person, blind in only one part of the colour spectrum, can adjust the use of colours in a way which is comfortable.

There might be someone in a department at work who is colour blind in part of the spectrum. Their PC might be set up with colours towards the red end of the spectrum. What is garish for one person might be perfectly restful for another.

It should be noted that emulators themselves have different characteristics. Powerlink™, the character-based emulator for

OfficePower™, allows the PC user to connect to a UNIX® server as if it were a VT220 terminal. It also provides facilities such as the transfer of files between the PC and the server. These files may contain graphics, word processed documents, spreadsheets and so on. Once on the PC they can be input to the appropriate PC based tool.

From the above we can see that, while we may be clear on what is Client-Server and what is not, the dividing line between the two is not so easy to define.

Client-Server systems commonly comprise four parts

- an intelligent workstation, probably a PC on the user's desk
- a local server supporting personal or workgroup functions
- a remote server supporting company-wide functions
- a network.

2.2 The Workshop Objectives

As already indicated, in an attempt to understand the full implications of Client-Server systems the customer commissioned Integration Services Division, part of ICL Client-Server Systems, to run a workshop covering 'the enhancement of existing systems by the addition of links to document imaging and other similar technologies'.

The remainder of this section follows the same sequence as that of the workshop, only one day of which was run on the customer site. Preparation was done at ICL Bracknell. In addition, the setup of communication links, and the loading of pre-requisite software, was performed before the workshop.

The starting point for the workshop was the customer's business with an analysis of their problems in business terms. What needed to be done to remain competitive? What needed to be done to retain end customer loyalty in an industry where there is much competition? Next it looked at a possible solution and the benefits that would arise from this approach. Finally, the workshop looked at what such an approach would mean in terms of future enhancements. In this paper, the appendix gives details of the code produced, setup constraints, equipment and software used.

2.3 The user's business problem

In this case the company involved was a UK travel firm. They operate in the UK and parts of France, Spain, Italy and the Netherlands. They own and operate some holiday camps and also hire out both boats and holiday cottages. To book a holiday you fill in a form at the back of a brochure indicating first, second and third choices and send it off to the booking centre.

The mainframe-based system had been in operation for several years and had served the company well. Whilst it has been successful, the customer felt that future extensions would require a different approach. The booking aspects of the system work well. Bookings have been supported by the mainframe application for some years. The problem arose when a prospective customer phoned up to enquire about the status of their particular holiday or to amend the details in some way.

The situation is that a customer will have a letter and probably a brochure in front of them and will be using expressions like 'the letter I wrote on the 12th', 'on page 7', 'the picture with the boy in the canoe' and 'the cabin on the left'.

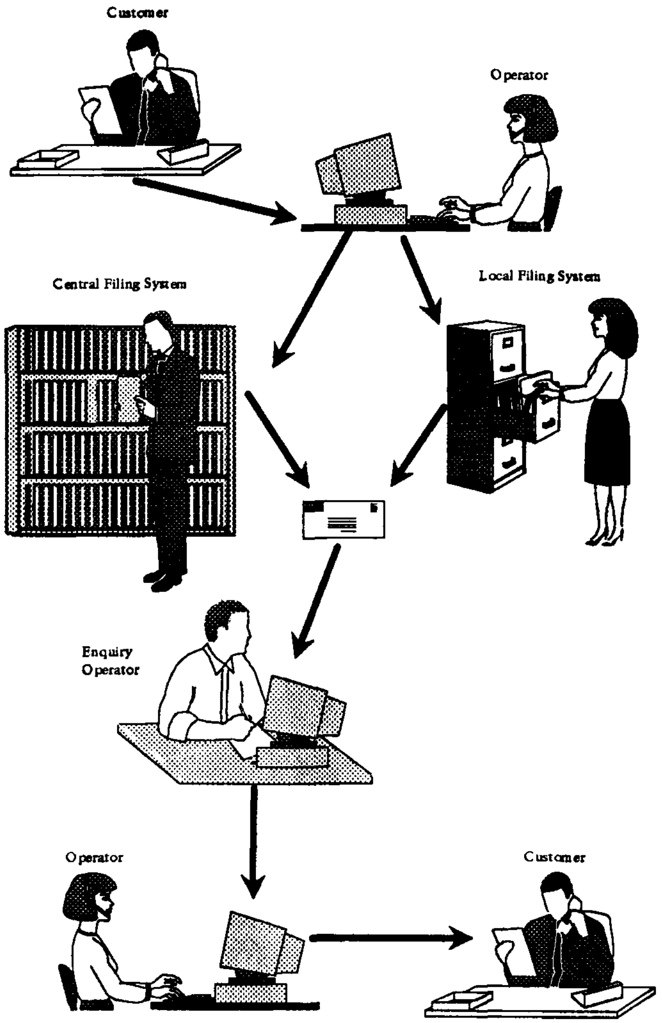


Figure 3 Customer enquiries re bookings - The 'Before' picture.

The operator on the other end of the phone has the benefit of a character based computer system. They may not have instant access to the appropriate correspondence. They may handle several brochures and may not have the actual page to hand. They will be forced to ask questions like

- 'what was our reference number?'
- 'what is the holiday start date?'
- 'what is your booking reference?'
- 'can you give me your post code?'

Even if the operator can find the relevant brochure, and the appropriate page within it, there is no direct link to the customer's original booking form and any subsequent correspondence.

In figure 3, the customer makes an enquiry by telephone. The telephone operator collects some details and passes the enquiry on to someone else. An initial enquiry may search local files in a filing cabinet. A more detailed search may have to be made of some central, larger filing system.

Finally, before the telephone operator contacts the customer, some details may have to be entered into, and cross checked against, a computer system. Only then can a response be made to the customer.

This approach has several disadvantages but two are immediately apparent. First there is a delay in answering queries because reference has to be made to local or central filing systems. When found, the details may have to be checked against a computer system, possibly not the one used by the original telephone operator. Finally there is the expense of contacting the customer either by phone or by letter, or both.

These factors potentially add up to delay, bad company image and expense.

2.4 The workshop prototype.

During the workshop, a prototype application was set up in which telephone enquiry operators were to have their terminals replaced by PCs. They were then to be given on-line access to the booking details on the mainframe, as before, but now with concurrent access to the brochure and correspondence details as well. To make it easy for them, the approach adopted was to provide direct automatic access to the brochure using the booking reference, or holiday reference, extracted from the mainframe system, or by direct input of the brochure page number. Similarly, automatic reference was to be made from the customer name, or booking reference, to the associated correspondence.

A brochure page often contains written details of four or five holidays. For each holiday there is at least one photograph. There is also a table showing charge rates by number of people and by start date for each holiday. Some pages also contain a map with overview directions.

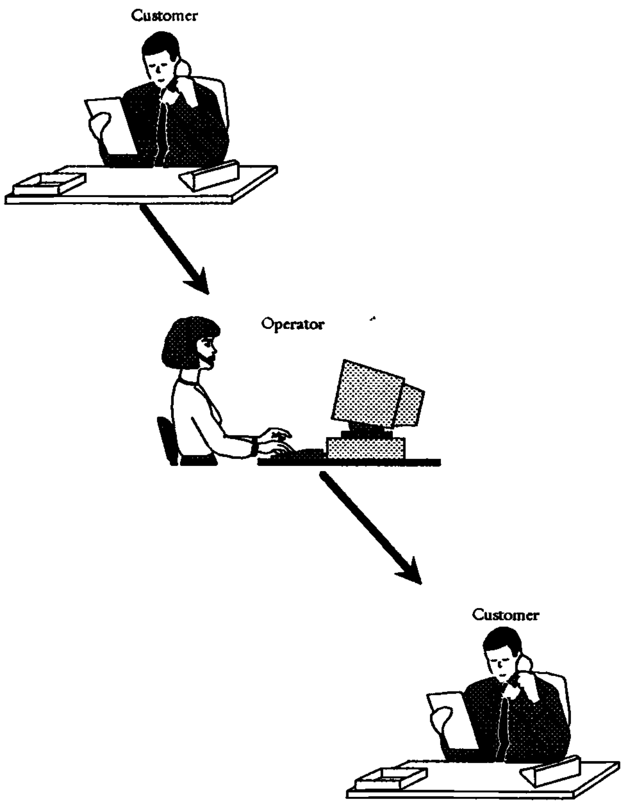


Figure 4 Improved Customer Service

With this on-line access to the brochure, a better impression can be presented to the customer. Queries may be answered over the phone quickly and using the same 'language' as the customer.

A simple prototype application implementing this approach was achieved in a day. The steps taken and the code used are to be found in the appendix to this paper. It should be remembered, however, that whilst a demonstration can be achieved in a day, a live implementation can take many months as it may well involve a change to company procedures, replacement of existing equipment, retraining etc.

2.5 Possibilities for future enhancement

The difficult part of this Client-Server environment is to establish a working link between that part of the application running on the mainframe and that part running on the PC. Having established this link then the same principle, as used for direct retrieval of document and colour images, can be used for sound and moving graphics.

This might lead our travel company to adopt some new approaches.

The combination of Windows 3 front ends, fast 486 processors, good graphics and a touch screen can lead to some innovative systems, see figure 5.

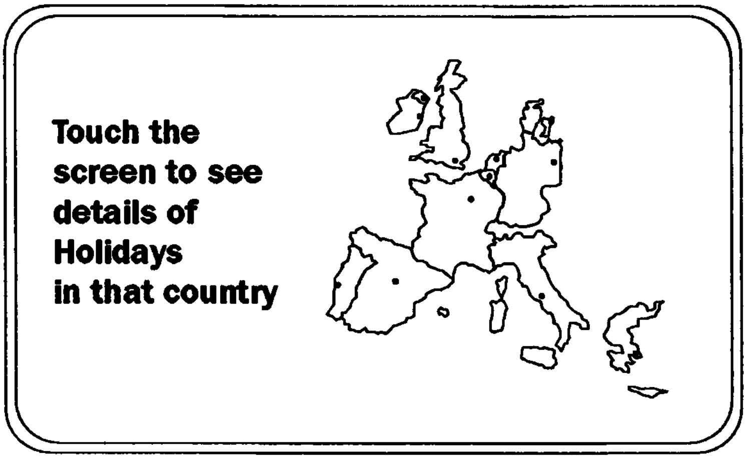


Figure 5 Possibilities for future enhancement

For example, having taken the trouble to produce the brochure in machine displayed form, e.g. on a CD-ROM, with document and photograph images, they might place touch screens in the concourses at major railway stations and shopping arcades. While waiting for the train, a potential customer might touch part of the screen and see a video clip, also on the CD-ROM, along with a sound bite 'selling' them on the merits of the French canals or the Cornish Riviera.

This touch terminal might even invite them to make their booking. Without the hassle of telephones, letters or delay, the prospective customer, via a quick swipe of a credit card, would know that the annual holiday in the South of France is assured.

3 Workshop

3.1 Outline of the workshop

The infrastructure for the workshop was set up in advance. This basically covered the procurement of a PC, the loading of appropriate networking software and the installation of terminal emulation software.

Before the workshop, advance copies of the latest brochures were sent to Integration Services, part of ICL Client-Server systems. Selected pages and photos were scanned so as to produce an image stored as an electronic file on computer disc.

Using the scanned images, a demonstration and presentation were built by ICL Integration Services to show the potential of Imaging. These made use of ICL's PowerVision™, Ref [5], for black and white document imaging, and FotoTouch Color, Ref [7], and Logitech ScanMan for colour photograph imaging Ref [6].

Certain principles were explained as part of the workshop. This covered the principles of windowing within Windows 3 and the use of Dynamic Data Exchange (DDE). A demonstration of interlinking of windows programs made use of Excel, Windows card file and Visual Basic, using DDE.

The prototype was then moved from an Integration Services PC, used for the demonstration, to the customer PC which already had a Terminal Emulation connection to the customer's ICL VME™ TP service. Several application screens within this TP service were enhanced as follows. The PC was set up to display several windows at once. Terminal emulation, to the mainframe, was active in one window and an image server in another window. When the customer details were displayed in character mode in the TE window, the operator was able to click on the image server window. Clicking on this image window caused the customer reference details to be passed automatically between windows and the relevant correspondence and/or brochure details page to be displayed.

These latter two steps were completed within the day allocated.

4 General guidelines for choosing a particular image system

Adding Image to an existing mainframe application is straightforward. However, before starting, certain choices have to be resolved.

- what sort of Imaging system is required?
- is it to deal with black and white or colour?
- will the system store and manage copies of documents or full colour photographs?
- what volume of images are to be managed?
- what will the purpose of the imaging system be?
- is the requirement merely to store records in a form in which they can be accessed infrequently or is regular fast access required?
- do people have to have concurrent access to the same image?
- how often do the images change?
- how mobile are the people?

The answers to questions such as these are outside of the scope of this paper. Suffice it to say that for the workshop two imaging products were used:

- ICL's PowerVision for black and white Document Image Management.
- The combination of FotoTouch Color, and a Logitech ScanMan for Colour Image management.

  Both products operate on the PC under DOS and Windows. ICL's PowerVision can also operate under UNIX.

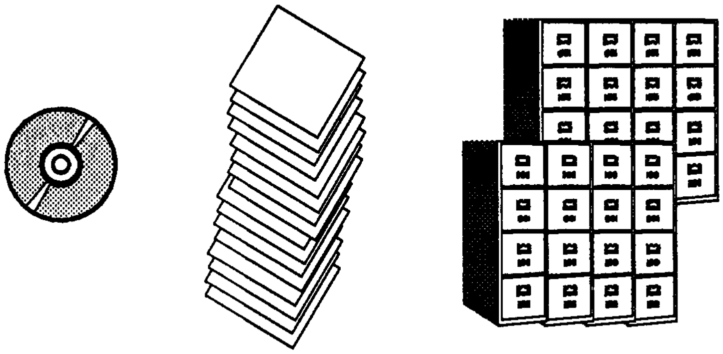


Figure 6 Some logistics of storage size.

Figure 6 shows one 12" Optical disc which can hold the same information as 200,000 A4 pages. These pages would themselves occupy 8 Four-drawer Filing cabinets.

A brochure with 250 pages would take 10 Mbytes if scanned in in black and white. The same brochure would take 20 Mbytes if the photos were included as photos rather than a scanned page, but still as black and white. Finally, such a brochure would take 100 Mbytes if scanned in as full colour, even though the text is black and white.

One 12" Optical disc could hold approximately 10 different brochures; plus around 4000 active customer files with booking details and additional correspondence and about 20,000 bookings where there was no additional correspondence.

4.1 Image enhancement to an existing mainframe application

The example code in the appendix is only valid for Windows based terminal emulators which support DDE. DOS terminal emulators (e.g. ICL's PowerLink) cannot interwork directly with Windows applications and therefore cannot be used for this purpose.

4.2 Design Steps

Four steps must be performed when writing a PC application which links a mainframe character application to an imaging system:

- decide how the images are to be indexed. Store the images using this indexing scheme.

- write code to pass the reference key from the terminal emulation screen to the image server.
- write code to transform this reference key to a file name. This file will contain the document name and view of the corresponding document. (Using the indexing method devised in step one above.)
- write code to call the imaging system software (PowerVision/PC, Ref [5], or FotoTouch here, Ref [7]) to display the picture, document or photograph using the file name and view found in the previous step.

The documents, photographs and pictures are scanned using an electronic scanner. An index is added before the images are finally stored on magnetic disc, or CD-ROM. Often the index is on magnetic disc and the image on CD-ROM. See figure 7.

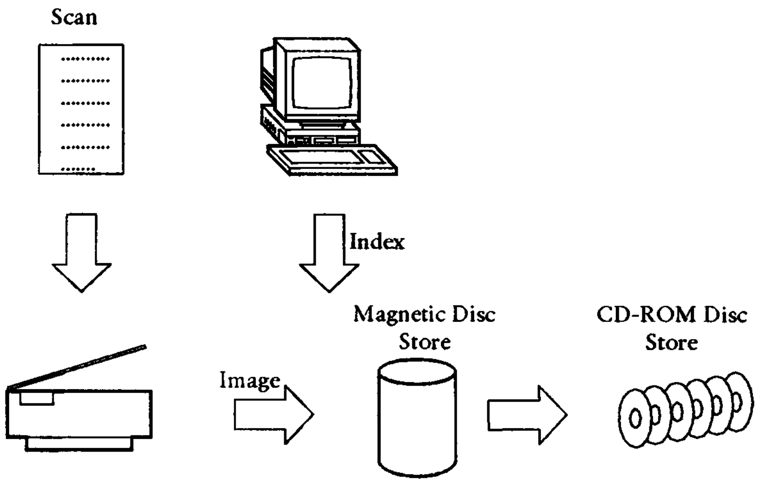


Figure 7 Steps in Indexing and retrieval.

4.3 Potential pitfalls

Many Windows 3 products allow the end user low-level facilities such as cut and paste. In a live situation you rarely want the user, the telephone operator here, to cut and paste manually. They may make a simple mistake - and then blame the system. Their blame would be justified as a good system doesn't allow mistakes. To avoid this you need a way of controlling all the PC operations automatically. This may constrain the end user to work only one way but it also ensures that errors are not made. For the workshop prototype Visual Basic was used as a controlling framework. Any one of a number of products could have been used such as Micro Focus COBOL, Gupta SQL Windows, C++, Windows Software Development Kit (SDK), Microsoft Access, Microsoft Visual Basic and Borland Turbo Pascal.

As an aside, one of the advantages of the COBOL approach is that many, or even most, mainframe TP systems are written in COBOL. If the graphics interface at the PC can be developed using COBOL then this maintains the discipline of using as few development languages as possible. There is, therefore, a minimum of retraining and support costs.

5 Conclusions and Recommendations

5.1 Conclusions

The workshop showed that, using the latest PC Tools, code to enhance the user image Graphics User Interface (GUI) is very easy. It is robust and can be used for so-called Industrial Strength applications. However, code to interface with other, existing, applications still requires a low level of detail. Also, for an operational system, there is not always that same level of control over the developed code as has been available on mainframe systems. Many PC-based development tools are aimed at the single developer, not a team of developers. They do not make use of a project dictionary with source control and documentation facilities.

Increasingly, however, dictionary-based products are appearing, such as that with the Network Designer product 'T.C.T.' and with the Micro Focus COBOL Workbench, Ref [2].

So the benefits to the end user are real. But we must not forget the benefits of control of the development process that we have learnt over the past 25 years.

5.2 Recommendations

It is worthwhile setting up a pilot using one from each of the two types of tools.

1. One of the new PC-based tools which have their own 'new' language and which are targeted at the PC as the complete development environment.

   Tools in this category include Microsoft Visual Basic, Gupta SQL Windows and Network Designers T.C.T.

2. One of the mid-range or mainframe-based tools that have been ported to the PC but which cooperate with the existing mid-range or mainframe system.

Products in this category include the Micro Focus COBOL Workbench, Ingres Windows 4GL and Oracle Forms.

This approach will provide valuable experience with the new tools. It also provides an incremental approach which doesn't require that the existing system be scrapped. Perhaps most important though is that it gives the Data Processing or IT department the chance to re-establish themselves as the visionaries within an organisation; those people who seek out new and better ways of doing cost effective business.

Acknowledgements

Many colleagues within the Systems Integration Group, Client-Server Systems and OPEN*framework* Division have contributed ideas and stimulated thoughts on this subject; I thank them all. Special thanks, however, go to Andy Laidler who assisted in the technical setup of the original workshops and who wrote the image indexing code.

Trademarks

UNIX is a registered trademark of UNIX Systems Laboratories, Inc., in the U.S.A. and other countries.

MICROSOFT is a registered trademark of Microsoft Corporation in the U.S.A. and other countries.

PowerLink and PowerVision are trademarks of International Computers Limited.

References

[1] Gartner Group Client-Server Computing Conferences May 1993.

The Gartner Group is a leading USA based IT Industry Watcher. They publish research and host conferences on IT related topics. They also act as consultants to major users and IT vendors.

[2] Micro Focus COBOL Workbench™; Micro Focus Dialog System™, April 1993.

[3] Microsoft® Visual Basic Language Reference, Version 3.0., 1993

[4] BRENNER, J.B. Client-server architecture, *ICL Tech. J.* Vol. 9 Iss. 1 pp. 3-17, 1994.

[5] ICL PowerVision/PC Programmer's reference manual. 54513/001 June 1993

[6] ScanMan Hand Held Scanner. Software user's guide 620430-02-EN July 1992

[7] ScanMan FotoTouch User's guide 620363-00-EN, July 1992

[8] 7561 Windows User's guide 15458/007 September 1992

Biography

Alan Beer

Alan Beer graduated from Leicester University in 1968 with an honours degree in Mathematics. His early career was spent working at Honeywell Information Systems where he was involved in the development and support of the disc based Bill-Of-Material-Processor. This was followed by several years working for two large IBM and Univac users on the installation and support of database systems.

He joined Logica as a Senior Business Consultant and, after a period of work in Continental Europe, joined ICL in 1979. Initially he worked on the development and implementation of ICL's IDMSX Codasyl database and TPMS, on-line TP Monitor. After two and a half years secondment to ICL Australia, introducing 4GLs into the Asia Pacific Division, he became Marketing Manager for QuickBuild, ICL's dictionary based 4GL development environment, and for Ingres, ICL's initial Relational Database on VME.

Alan is currently Business Development Manager of the Business Team in *The* SOLUTIONS *Centre* within OPENframework Division. This unit undertakes workshops and sales training on all aspects of Client-Server implementation and support and gives seminars on Client-Server based on the practical experience gained.

Appendix A

A.1 Glossary of terms

DDE. Dynamic Data Exchange

Most PCs run DOS and Windows 3. DOS and Windows 3 together form a single tasking environment. This means only one task, as viewed by the user, can be active at a time on the PC. In the examples above we require to have at least two tasks active (or at least appearing to be active). To achieve this we can make use of Dynamic Data Exchange. This is a Windows feature which allows data to be passed from one Windows 3 'window' to another.

DT.Dumb Terminal.

A terminal dependent on a mainframe computer for its operation. Often green characters on a black background. Sometimes referred to as 'legacy terminals'. Similarly the character based applications which reside on the mainframe are sometimes referred to as legacy applications.

TE. Terminal Emulator

This is a piece of software running on the PC which allows it to communicate with the mainframe 'as if it were' a mainframe driven terminal. Common emulators are 7561 to communicate with an ICL mainframe; 3270 to communicate with an IBM mainframe; and VT220 to communicate with UNIX and DEC VAX environments.

A.2 Workshop details

The following is provided for completeness only. It gives an indication of the types of routines required and their relative simplicity. The following examples relate to the use of Visual Basic but, as indicated in the text above, other Windows based products could have been used.

A.2.1 Design and Coding of the prototype example

The example given below has been written in Microsoft Visual Basic.

A similar procedure would produce the same results in any application development environment which supports DDE (Dynamic Data Exchange - the method for communicating between Windows applications).

A.2.2 Decide how images are going to be indexed.

The prototype program had an indexing file (a straight text file) which contained lines with page numbers (or holiday references which contained page numbers within them) and corresponding file names.

For Example:

```
Page File
----- --------
084.1 p084.tif
084.2 p084.tif
```

This maps the page number, 84, and holiday reference, holiday 1 on page 84, onto the image file, p084.tif.

Each image (page) in this example contains more than one holiday, and so it is desirable to "zoom in" on the selected holiday.

This is done using "views". In ICL's PowerVision it is possible to save the required positioning and magnification of the image as a "view" and to recall it by name. For the prototype example, the view used for each holiday number is named as the holiday number itself. For example, the image for 084.1 is in file p084.tif (from the above table) and has view 084.

On a live system, since there are only 4 or 5 holidays per page, it would be acceptable to define 4 'Views' TLH (Top Left Hand), TRH, LLH and LRH (Lower Right Hand).

The view for the end user will be of 'One System', where one window contains information from the mainframe. Another window may contain a photo or some other image from the brochure. Another window may contain part of an Excel spreadsheet say with financial details.

A.2.3 Obtaining the holiday or booking reference number automatically from the Emulator screen.

The method for doing this is dependent on the terminal emulator in use. However, there are three general principles for extracting portions of the screen from a terminal emulator:

In the following example, code comment lines start ***

a. Open a DDE link to the Terminal Emulator and execute a DDE command which puts a portion of the screen onto the clipboard. DDE commands are specific to the emulator in use and are normally described in the emulator documentation, e.g.

```
Text1.LinkMode = 0

*** Make sure Text1 is not linked to anything.

Text1.LinkTopic = ....

*** Specifies the other end of the link. (See emulator manual

*** for the required link topic for your emulator. Since we were

*** using win7561 the topic was win7561|win7561).
```

Text1.LinkMode = 2

*** Activate the link.

Text1.LinkExecute "Command"

*** Perform the operation.

Text1.LinkMode = 0

*** Unlink from the emulator.

In this case, "Command" would probably be of the form:

gettext(x,y,a,b)

Which would get a region of text starting at column x, row y, and a characters wide, b characters deep and store it for use by DDE.

It is then necessary to get this text from DDE into the application.

b. (This is the method used by vt220 Plus and win7561 Windows.) In the terminal emulator configuration window, define a region of the screen to be associated with a name.

(e.g. for vt220, edit the .220 configuration file to include the line

DDEn = '*name*', x, y, a, b

This associates the name "*name*' with the screen region starting at x, y and being a wide and b deep)

Thus, whenever an application uses DDE to connect to the emulator using the name defined above, the emulator will send the contents of that portion of the screen to the application.

A.2.4 Displaying an Image on the Screen.

The following assumes the display of a PowerVision image. A similar set of code would be used for other DDE enabled products. It is necessary to first open a DDE connection to the Work Station Manager (the part of PowerVision which deals with the DDE communications), then to send the commands required to display an image.

Text1.LinkMode = 0

*** Open up the DDE connection in file mode

Text1.LinkTopic = "Wsm|file"

Text1.LinkMode = 2

Text1.LinkExecute "[openwindow(local1,1,Display Window)]"

*** Open a window in which to display the image

Text1.LinkExecute "[displayimage(c:\filename,t)]"

*** display the appropriate image

This is a very simple way to display an image, for all the options available see the PowerVision Programmers Reference.

PowerVision can store images in TIFF (ending .tif) or PowerVision Internal, CCITT Fax 4, format (ending .pvi).

FotoTouch can store images as Bit mapped (BMP), TIFF (Compressed or uncompressed), TIF, PCX or JPEG.

A.2.5 Other subroutines used in the workshop.
The following are added for completeness. An explanation follows each piece of Visual Basic, Ref [3], code.

```
Sub Do_Movement ()
com$ = "[setview(" & Text1.Text & ")]"
MsgBox "Text1.Text
*** Diagnostic only. Show what page and view we are looking for
Text1.LinkExecute com$
End Sub
*** This subroutine, above, is called to focus on the appropriate
*** view of a page.
Sub get_filename (temp$)
        Open "c:\directory\pages.doc" For Input As 1
        temp$ = "dummy string"
        Do until (Left$(temp$, 6) = Text1.Text Or EOF (1))
            Line Input #1,temp$
Loop
If (EOF (1) And Left$(temp$, 6) <> Text1.Text) Then
        MsgBox "Page number not found."
End If
temp$ = Right$(temp$, Len(temp$) - 7)
End Sub
*** This matches the page name to an image that has been stored
*** previously
Sub Command1_Click ()
*** Get Page number from the emulator (win 7561)
        Text1.LinkMode = 0
        Text1.LinkTopic = "win7561|win7561"
        Text1.LinkItem = "R3C16:R3C23"-
*** Here the field is on screen at Row3 Col 16 to Row 3 Col 23
        Text1.LinkMode = 2
        Text1.LinkRequest
        Text1.LinkMode = 0

*** Open The PowerVision Window
        Text1.LinkMode = 0
        Text1.LinkTopic = "Wsm|file"
        Text1.LinkMode = 2
        If (Err <> 0) Then
            MsgBox "Cannot DDE connect with WSM"
        End If
        Text1.LinkExecute "[openwindow(local1,1,Display Window)]"
        If (Err <> 0) Then
            MsgBox "Open failed"
            End
            End If
*** Translate Page Number into FileName
    get_filename temp$
*** Tell PowerVision to open the file
Text1.LinkExecute )\ & temp$ & ",t)]"
*** Zoom in on the page/Holiday in question
        Do_Movement
        Close
        AppActivate "Display Window"
        AppActivate "Display"
End Sub
Sub Command2_Click ()
```

```
If Text1.LinkMode = 0) Then
        MsgBox "Error, An image is not active"
        End If
Text1.LinkMode = 0
End Sub


Sub Form_Load ()
        T= Shell(C:\iclpv\wsm.exe",4)
End Sub
*** Load up workstation manager first time round-
```

A.3 Software and Hardware required for the prototype described here

In constructing the prototype the following were used:

1 386/486 PC, for speed when working with graphics, with network connection to the mainframe application being used.

2 A Terminal Emulator which supports DDE.

3 Software capable of displaying the scanned images. PowerVision PC and FotoTouch were used here.

4 Access to a scanner to scan the images into a machine readable form.

5 A development language which supports DDE.

For running the application, the first three items are required plus the application itself. Many of the PC development languages have a run-time subset. This subset is of lower cost and uses less machine resources. However the run-time subset does not support development by the user. Either a full development environment or a runtime subset must be available for the application to run.

An accelerator card can be used to improve the speed of graphics. Where an operator is constantly referring to high quality images then such a card is highly desirable. The workshop was run on 486 PCs without the benefits of these cards. A 386 PC without acceleration would be acceptable only for infrequent usage.

Legacy systems in client-server networks: A gateway employing scripted terminal emulation

Paul Duxbury
Client-server Systems, ICL, Kidsgrove, UK

Abstract

A successful commercial client-server architecture cannot ignore the vast legacy of existing applications on a wide assortment of platforms and networks, developed over decades. Some can be re-engineered but, for many, the only practical mode of access is from a native terminal. A generic server which enables such applications to be sponsored as network services would be highly desirable. Such a gateway server has been produced, initially for the ICL TP Manager system but intended to be reusable in other client-server architectures. It works by scripted terminal emulation.

This article describes the technology from three angles: the image presented to potential customers; a more formal set of engineering requirements; and the actual implementation. It ends with an evaluation of the gateway's effectiveness, particularly the ease of scripting. Since adaptability to existing legacy systems, protocols, client-server architectures, ready-made components, etc., is a key factor, the whole project may be considered an extended exercise in bottom-up design.

1 Functional description

This first section introduces the concept of a Scripted Terminal-emulation Gateway, whose purpose is to help integrate legacy systems into newer client-server networks. It is deliberately styled as a *sales pitch* and might be considered the top-level marketing requirements for such a product.

Subsequent sections probe beneath the gloss to discuss some more formal engineering requirements, an actual implementation, and its ultimate effectiveness. The intention is to provide a rounded description from various viewpoints.

1.2 Legacy software is VALUABLE software

Modern client-server solutions, distributed application platforms and the like are all very well for *green-fields* developments, but what about the vast legacy of traditional software? Take, for example, the billions of lines of mainframe COBOL which have been written over decades, and the data they control. There are also plenty of examples in the asynchronous world of departmental systems (e.g. UNIX®).

The ability to tap into this rich legacy from within the emerging distributed application architectures would provide benefits all round (figure 1).

End users:	- Preserve their investment - Achieve more phased migration
Client-server vendors:	- Benefit from ready-made servers - See fewer migration obstacles
Original products:	- Benefit from new lease of life

Figure 1 Summary of benefits

However, despite these benefits, the cost of changing legacy software to fit the client-server model is often prohibitive - a barrier between cultures and ultimately a barrier against progress (figure 2).

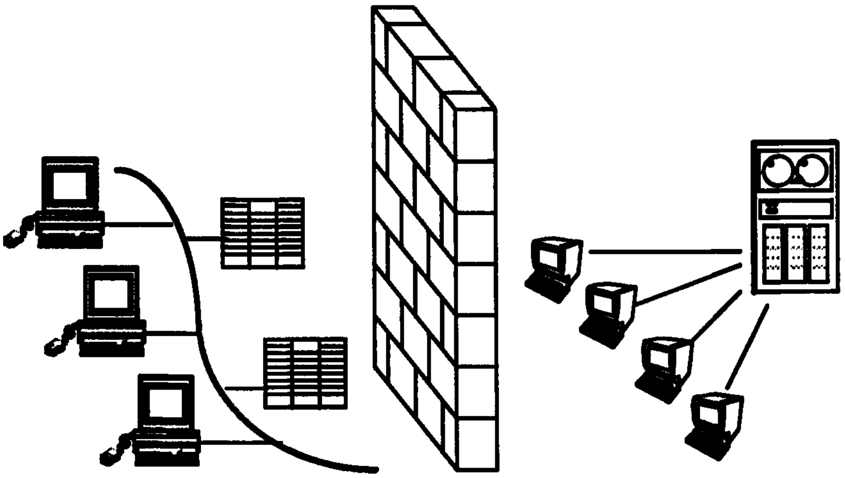


Figure 2 Alien cultures?

1.3 Scripted Gateway provides a *time tunnel*

What is needed is some magic *time tunnel* between the new client-server and the old centralised system architectures. This would make older but nevertheless valuable programs appear to new clients as modern networked service providers. New clients, on the other hand, would appear to these programs as old-fashioned terminals. So both would be happy!

The Scripted Gateway provides just such a time tunnel. Its basic function is illustrated in figure 3.

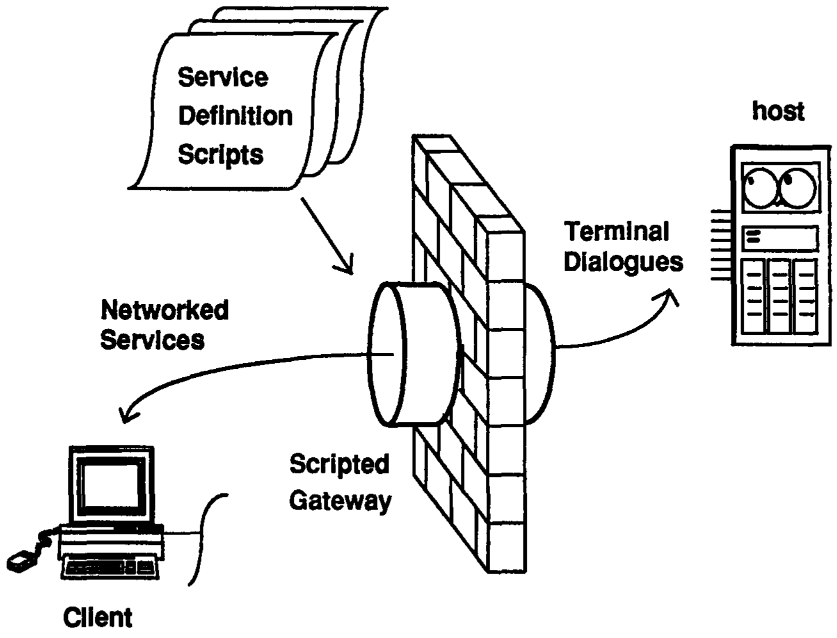


Figure 3 Scripted Gateway *Time Tunnel*

The facilities offered by the host program/system are re-defined by the gateway as networked services, in a form appropriate to the specific client-server architecture in use, i.e. as Remote Procedure Call (RPC) stub libraries, ICL TP Manager (TUXEDO®) servers or the like. These are advertised on the client network in the normal way. The services themselves are implemented within the gateway as scripted terminal dialogues with the target host program.

The client and host are shown here as a PC and a mainframe application respectively, but they could equally well be just two different programs in the same UNIX box, or some other configuration.

1.4 Traditional UNIX programs can act as network servers

Take as a first example a traditional UNIX application which expects to talk to dumb asynchronous character terminals. It can, of course, be accessed from a PC by direct terminal emulation, and windows systems even offer some limited integration capabilities like cut and paste.

But the presentation style of the target program, often archaic, remains visible to the end user, as do the (manual) integration procedures. Moreover, there is limited scope for true value-added integration with other applications, local or remote.

By contrast, introduction of the Scripted Gateway into the server system offers the host program's services to clients in a much more convenient form, perhaps using some type of RPC, or TP Manager's /Workstation product. Indeed, the interface is then indistinguishable in style from that of purpose-built servers.

In the other direction, the Gateway talks to the target host program by terminal emulation through a UNIX pipe or pseudo-terminal - as a bonus, this reduces the heavy load on the network normally incurred by character traffic. See figure 4.

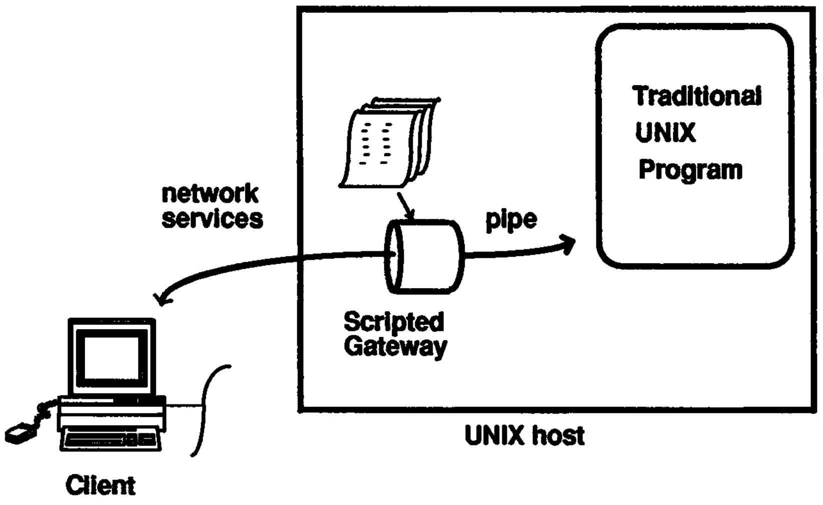


Figure 4 New life for an old UNIX program

The scripts might be produced by:

- the client vendor to increase market penetration;
- the original program vendor to prolong its life;
- a customer or service outfit to achieve custom integration;
- a third party as value-added products in their own right

1.5 A UNIX server can front-end a mainframe host

In many situations it is not feasible to run the Scripted Gateway in the same system as the target program - for example, the system may be a non-UNIX mainframe, or it may be outside the solution-provider's control (perhaps being a public service).

Nevertheless, the Gateway can still be employed, by introducing a UNIX system between the client network and the host. Such a system may well exist anyway, for departmental applications or as a concentrator. The situation is indistinguishable on the client side, whereas on the host side an external comms link to the host is needed rather than just a pipe. (See figure 5.)

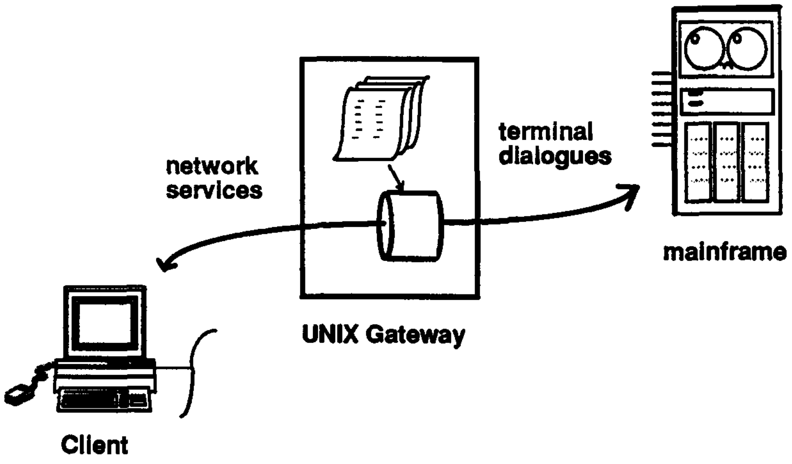


Figure 5 Front-ending a mainframe

The UNIX system may be positioned physically close to the client network (e.g. if it is a departmental initiative), or close to the host system. The latter offers particular benefits where the host operates an asynchronous terminal protocol, since the traffic is then removed from the corporate network - especially relevant for Wide-Area Networks (WANs).

1.6 TP Manager and Dialogue manager integrate legacy systems

The integration possibilities of the Scripted Gateway are widened still further when used in conjunction with a distributed transaction processing system, like ICL's TP Manager product. This is based on the TUXEDO system and allows multiple local and remote transactional services, typically databases, to be integrated within its client-server architecture. The inclusion of the Scripted Gateway extends this capability to other applications which might provide useful services to the application and its clients (figure 6).

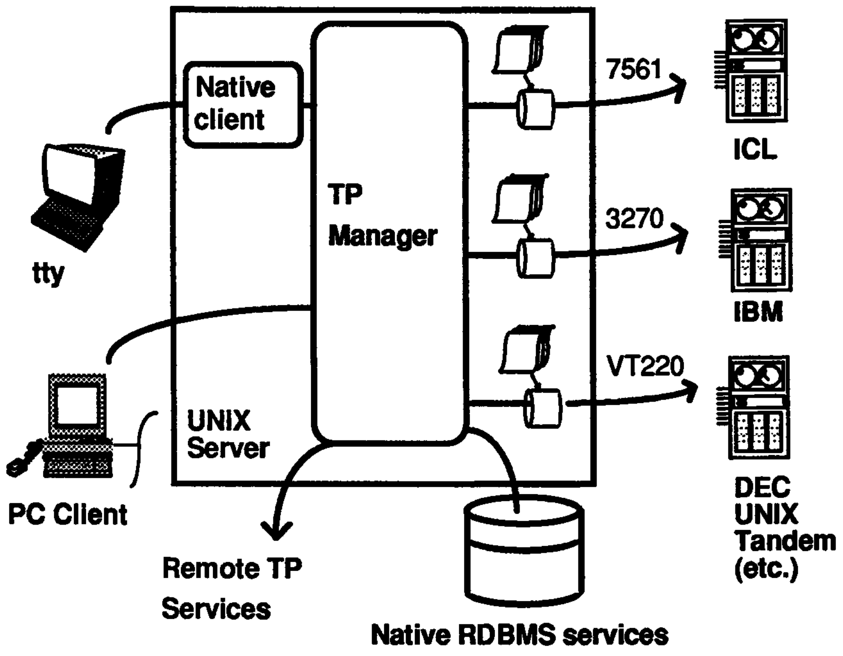


Figure 6 Integrating legacy systems with TP Manager

Multiple terminal sessions and multiple terminal protocols (VT220, 7561, 3270) may be employed within the same application using separate gateway instances orchestrated by TP Manager. In addition, and where appropriate, many clients may share a smaller number of host sessions, thereby reducing host overheads and network costs.

Integration can also be achieved within the client itself, using ICL's Dialogue Manager technology, which is frequently used in conjunction with TP Manager to achieve multi-stage integration and modularity.

ICL intends to supply a ready-made Scripted Gateway Server for use with the TP Manager product. In addition, the gateway could be made available as C libraries to enable servers for other client-server architectures to be built.

1.7 Scripting harmonises systems administration

As a possible example of systems integration using scripted terminal emulation, consider the problem of administering multiple, heterogeneous IT platforms.

Over time most companies end up with a mix of IT equipment, often requiring a corresponding mix of IT skills, both to use and administer. While this may be tolerable where the systems are to be used for different purposes by different people, there are, nevertheless, important situations, like administration, where this is not so.

From the point of view of administration, most systems provide essentially the same set of operations - adding users, managing filestore and so on, but in completely different ways requiring different skills and perhaps staff. Even worse, in some cases users need to be represented in a number of IT platforms and services simultaneously, and in a co-ordinated way.

Modern approaches to this problem are usually *object oriented.* The topic area (here systems administration) is described by a single set of abstract objects (users, volumes...) and services on those objects (add, delete, mount...), common across all platforms to be managed. This simplifies the task of administrators and client-software developers, but expects each platform type to interpret abstract commands in its own terms.

Where such direct support is not available, Scripted Gateways can help by turning abstract requests into appropriate terminal dialogues with the target platforms. A standard set of script interfaces is defined (the abstract service definition), and then an implementation is provided for each platform type. This is illustrated in figure 7. (An actual realisation might also employ TP Manager and/or Dialogue Manager.)

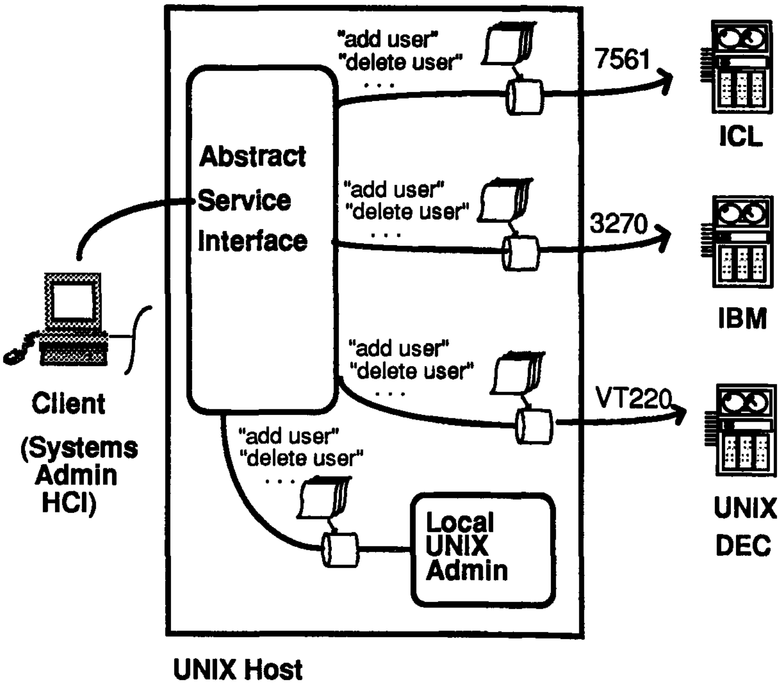


Figure 7 Administration of heterogeneous systems

1.8 Scripted Gateway is flexible

To fulfil the promises made of it this far, it is clear that the scripted gateway needs to be flexible in two dimensions: it needs to fit into a variety of modern programming architectures and it needs to cope with the widest range of legacy software.

To see how these are achieved we need a closer look at its structure. The gateway is composed of several levels and flavours of functionality. You choose as little or as much as you need (see figure 8).

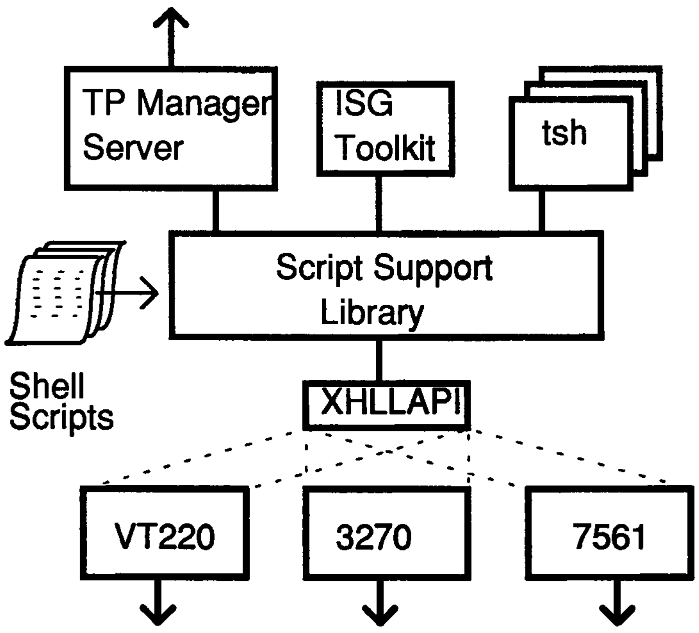


Figure 8 Take just as much as you need!

At the top level are some ready-made programs, for example, a TP Manager Gateway Server, and the **tsh** set of commands. These just require you to write scripts. Servers for distributed programming environments other than TP Manager are anticipated, as are standard sets of scripts for popular legacy programs.

If the ready-made gateway programs are not appropriate, then you can build the underlying script support library directly into your own program. This program might itself provide an embedding into some other distributed programming environment.

Finally, you can elect to drive terminal sessions directly from your own program, without scripts, using the virtual terminal facilities of the XHLLAPI library.

Accommodating a wide range of target legacy applications is done by providing a choice of XHLLAPI terminal emulations over a variety of media: currently VT220 is scheduled for release, a 7561 prototype exists and a 3270 version is intended shortly.

2 Requirements

The above described the scripted gateway technology as it is intended to appear to an end customer. The present section lists the requirements from an engineering perspective. Individual *requirements* are labelled **Rn.n** for cross reference in the subsequent implementation section.

2.1 General Requirements

R1.1 The gateway is required to offer a high degree of flexibility in respect of target host, protocol, language, client-server infrastructure etc.

R1.2 Existing components should be re-used wherever possible, and new components should be designed with re-usability in mind.

R1.3 Where relevant, open standards should be adopted.

2.2 Client-server infrastructures

There are, of course, many client-server infrastructures which could benefit from a scripted terminal-emulation gateway. Our primary requirement was:

R2.1 a gateway server for ICL's TP Manager product (a UNIX-based distributed Transaction Processing system incorporating the TUXEDO infrastructure from UNIX Systems Laboratories, Inc.).

However, it was quickly recognised that the bulk of the gateway functionality was independent of the actual distributed programming environment. Therefore, a secondary goal became:

R2.2 Re-usability within other client-server environments.

R2.3 Re-usability in stand-alone applications.

2.3 Target hosts

Clearly there is an immense diversity of legacy systems and software. Any capability for integration ought, therefore, to accommodate the widest range and be readily extensible. For the gateway this requirement was interpreted as a threefold need:

R3.1 for multiple terminal emulations, preferably *plug-in* modules

R3.2 for appropriately configurable communications

R3.3 for support of various application HCI styles, e.g. screen-mode (including templates), scroll-mode, asynchronous, and hybrids like VME MAC which can output templates in scroll-mode.

As with other gateway features, re-usability was a key factor:

R3.4 Existing terminal emulations should be adopted where feasible.

R3.5 Where we would need to produce our own terminal emulations, we ought to capitalise on the common aspects of such emulations.

R3.6 Our own terminal emulations ought to be re-usable outside the gateway itself.

2.4 Levels of service

The gateway works by providing client programs with a terminal emulation capability assisted by scripts. This leads to the question - how best to split functionality between client programs and scripts. Flexibility in this respect was required in order to accommodate differing circumstances of motivation, host HCI style, host stability, skills availability, network bandwidth and so on. For example:

- Where a host session can be redefined functionally as a set of distinct high-level operations, these can often be implemented entirely as scripts, giving an image identical to that of native services.
- In other cases, scripts may just be used to encode screen formats, leaving session control to the client but insulating the latter from actual screen layouts, in the manner of template libraries.
- In still other cases, the client program may require direct access to the virtual screen and keyboard metaphor, without any scripting and insulated only from the actual terminal protocol.

These are summarised as:

R4.1 The gateway must provide clients with an appropriate choice of service levels (images of the target host session) ranging from essentially transparent to completely abstract.

R4.2 Mixing levels of service must be possible within a single session.

2.5 Scripting Language

Languages are largely a matter of taste, and there are also performance and availability considerations surrounding implementations. Therefore, while needing to include a scripting language, it was also recognised that it ought to be custom-replaceable.

The bundled language itself was expected to be flexible in its own right and not to impose a continual development load for extensions. These requirements are summarised thus:

R5.1 The scripting language should be custom-replaceable.

R5.2 A suitable language ought to be bundled or be readily available without further cost.

R5.3 The bundled language must be readily extensible.

R5.4 The bundled language ought to be open and familiar.

R5.5 An existing language would be preferred, for reasons of expedience (in both specification and implementation).

2.6 Performance

For obvious reasons, high performance was not expected to be an attribute of the gateway, i.e. when compared with distributed programming techniques for host integration. It is intended for use when no such distributed programming capabilities are available.

However, it was required that solutions based on scripted gateway technology could be customised for optimum performance within its limitations, where feasible. Pluggable script languages might be one instance, where a compiled language could be substituted for an interpreted language.

R6.1 Selected strategic components ought to be custom-replaceable for performance enhancement.

On the other hand, in some respects new client-server applications based on the scripted gateway might be expected to perform better than direct terminal access to the host because much of the purely interactive work, such as form-filling, on-line help, etc., can be delegated to the client platform.

Furthermore, for certain specialised but commercially important configurations, of a transactional nature, it ought to be possible to allow many clients to share a smaller number of host sessions in the same way that a transaction monitor like TP Manager might allow sharing of native database sessions.

R6.2 Where appropriate, host sessions should be serially re-usable between many clients.

2.7 Security

Host security needs to be maintained. Requirements will generally vary with specific security policies but, in general, it is undesirable to require passwords to be included in scripts. Also, end-user accountability at the host is often required. These are complicated by factors like the need for automatic recovery of host connections and the sharing of host sessions among multiple clients.

R7.1 Scripts should not be required to hold passwords in clear text.

R7.2 End-to-end authentication and accountability should be possible as required by local and host security policies.

R7.3 Automatic re-login should be possible, without end-user or administrator intervention, where the reason is a simple host session break.

2.8 Server platform

We accepted that, in the first instance, implementation would be confined to UNIX server platforms. This happened to be the initial product requirement and is considered the most open server platform currently available. DOS/Windows PCs tend to be well covered by single-user terminal-emulation tools.

R8.1 The server should operate on UNIX platforms;

R8.2 The server should port readily to other POSIX conformant platforms.

2.9 Tools

Recognising the diversity of host systems and applications, tools are essential to enable scripts to be generated. The following requirements were identified:

R9.1 Generation of scripts should be largely automated by enabling the capture of interactive host sessions.

R9.2 As well as off-line utilities, *on-line tools* should also be provided, for example, the ability to view the host session in real time as it is being operated by clients.

R9.3 Tools should support all levels of service (i.e. from transparent to abstract), and all configurations of the technology.

R9.4 Tools should faithfully reflect the target environment, e.g. by re-use of actual product components.

R9.5 End-user tools should double as test programs, diagnostic utilities, and validation tools for the gateway itself.

3 Implementation

This section outlines the design of the resulting scripted gateway product set.

The above requirements largely reflect the need to fit into an existing environment. Therefore a bottom-up approach was adopted and will be so described. Cross-reference is made to the requirements individually identified in Section 2 (as R1.1, etc.).

Naturally, the first step was to establish the existence of any existing commercial products with the required characteristics. There turned out to be several products in the field but covering only different subsets of our requirements - some are strictly PC based; some are restricted to specific protocols like 3270; some focus on providing graphical user interfaces to single applications, rather than turning the applications into general abstract services for subsequent integration. Hence, we embarked on a bespoke development, though with the objective of re-using existing components wherever possible (R1.2).

3.1 Structure

Most of the Scripted gateway functionality is encapsulated in a layered series of dynamic libraries (also known as UNIX Shared Objects) for the C language. These are completely unaware of the specific client-server environment which pertains, if indeed there is such an environment. (R2.2) The structure is illustrated in figure 9.

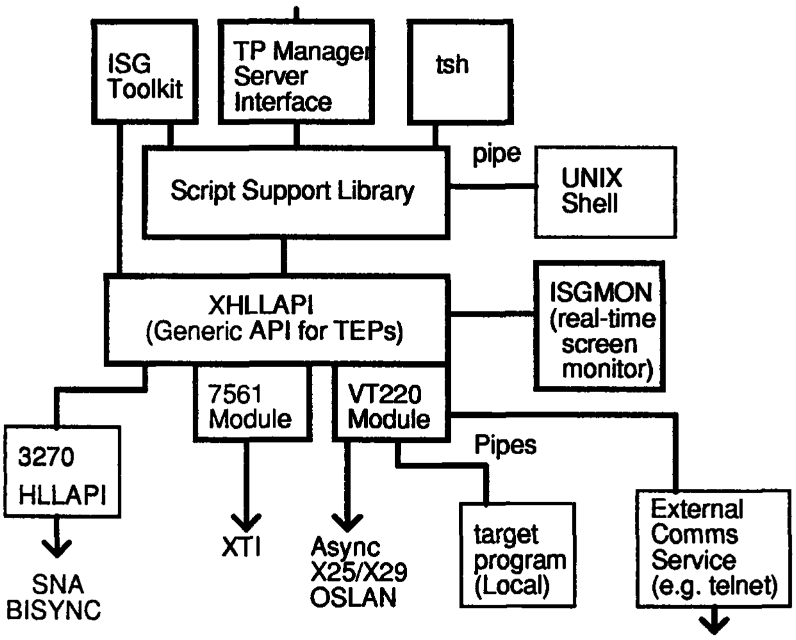


Figure 9 Scripted Gateway Structure

The resulting TP Manager gateway server was then written as an application making use of these libraries and itself merely providing the TP Manager (TUXEDO) server embedding (R2.1). Tools use the same libraries.

This approach has the following benefits:

- Other client-server platforms can readily be accommodated by building servers based on the same set of libraries (as a trivial example, a stand-alone UNIX command, **tsh,** has been produced to enable otherwise unsuitable, possibly remote programs to be included in UNIX pipelines);
- The layered libraries offer different levels of support to client software as required - ranging from a direct virtual terminal image to an abstract service call image implemented entirely as scripts (R4.1, R4.2);

- Dynamic libraries are not linked until run-time, so they can be substituted by alternatives (R6.1);
- Tools use the same code as the end-product and so see the host session in exactly the same way (a bit like a single-lens reflex camera) (R9.4);

Since TP Manager operates on UNIX platforms, all gateway components were developed for UNIX, using standard libraries (XTI, etc.) wherever possible to ease porting (R8.1, R8.2). The services offered are, of course, available to PCs via the TP Manager /Workstation option, as well as to other UNIX systems.

3.2 TEP libraries

At the bottom of the stack of interfaces is a set of Terminal Emulation Program libraries (TEPs). These provide just the protocol-specific aspects of terminal emulation and, as such, are not complete in themselves, so they are not directly usable outside of the gateway. They are implemented as separate dynamic libraries so they can be substituted one for another at run-time (R3.1). In fact, multiple instances can also be used concurrently.

3.3 XHLLAPI

The lowest level library available to users is known informally as XHLLAPI (The X prefix does not imply any standard). It is a generic application programming interface for terminal emulation and is usable as a free-standing library (R2.2, R2.3, R3.6).

The XHLLAPI interface is modelled on EHLLAPI, an equivalent for PC-based 3270 emulators of which an example is included in ICL's 3270 Client for Windows (R1.3). Use of a common existing interface was intended to enable existing terminal emulators, particularly 3270, to be substituted (R3.4). It might also enable XHLLAPI to be substituted for EHLLAPI in existing 3270 applications (R3.6).

The interface provides a virtual terminal metaphor with a rich set of functions by which client programs can simulate keystrokes, read/write/search the virtual screen contents, synchronise with host output and so on.

Extended features of XHLLAPI include support of non-3270 emulations (via the plug-in TEPs), concurrent sessions involving the same or different TEPs, asynchronous protocols, and specific scroll-mode support (R3.3).

From a practical point of view, XHLLAPI embodies all the functions common to all the terminal emulators. Thus over 90% of the code is re-used between TEPs. (R3.5).

3.4 Host communications

Host communications are implemented by XHLLAPI on behalf of the individual TEPs. It includes interfacing via the standard UNIX transport service interface (XTI) to OSLAN, TCP/IP and X25, including the ability to push extra protocol modules onto the protocol stack, for example, PAD emulation. Direct UNIX devices, like RS232, are also supported, as are other UNIX files including named pipes. (R3.2)

As a catch-all for situations where the above are insufficient, XHLLAPI can be made to pipe its host communications through a configurable UNIX command. This runs as a separate process and can be custom-written if necessary. In fact, asynchronous access over LAN is often piped through telnet (R3.2).

3.5 Script Support Library

Whereas XHLLAPI offers its users a wide selection of low level functions like 'read x characters from the virtual screen starting at position y', the library above it offers a much simpler, higher level capability based on scripting. Again, it can be used as a free-standing library (R2.2, R2.3), possibly in combination with XHLLAPI (R4.2).

The main function has the unassuming title 'run_script', but it is best considered as providing the ability to execute a named service with associated arguments (named or positional). It is obviously well suited to supporting client-server paradigms like Remote Procedure Call.

As the name Script Support Library implies, the services are implemented by running correspondingly-named scripts. The complexity of these scripts determines the level of service offered to the client program (R4.1). Two common approaches are to:

- Implement an entire abstract service as a script, e.g. 'Deposit amount x to account y', or 'place order for x, to be delivered to y by date z' and so on;
- Capture just screen layouts in scripts, leaving the actual dialogue sequencing to the client - e.g. 'send form DEPOSIT with fields ...', and 'Receive form NEWBALANCE into variables ...'.

3.6 Scripting Language

As a bundled scripting language the UNIX Shell is employed. Though not elegant (again a matter of taste) it does have the advantages of being readily available, widely used, and an open standard (R5.2, R5.4, R5.5). It is also readily extensible through the use of shell functions and UNIX command pipelines (R5.3).

Shell scripts are executed by the script support library, on demand, by executing the Shell command as a separate process and linking to it via UNIX pipes. The pipes carry a simple textual protocol enabling parameters and results to be exchanged and low level (XHLLAPI) services to be offered to the scripts. These are hidden behind a set of shell functions, such as:

```
tsh_read <variable> <row> <col> <size>
```

which reads a string from the virtual screen into a named shell variable.

This arrangement means that any UNIX command could be substituted for the UNIX Shell, and therefore any other language system capable of reading and writing simple UNIX character files can be employed. Likely examples are variants of Shell (C Shell, Windowing Korn Shell, etc.), and compiled C programs (R5.1).

Most of the services offered to scripts by the support library are based on XHLLAPI operations. An exception is the ability to invoke pre-stored passwords held by the library in order to avoid the need for them to be coded in the scripts themselves (R7.1). Password storage is volatile, passwords typically being supplied initially by clients as arguments to a login service (script), and subsequently available to the gateway for automatic re-login following loss of the host session (R7.3).

The modular structure of the gateway itself and the scripting mechanism, as outline above, is considered to fulfil the basic requirements for flexibility and re-use (R1.1, R1.2).

3.7 TP Manager Gateway

An example of the use of the above libraries is their embodiment as a TP Manager server, known as Intelligent Server Gateway (ISG), which happened to be the initial requirement (R2.1). The extra functionality needed is quite small, basically just the ability to translate service calls and data structures from the format demanded by the particular client-server infrastructure (in this case TUXEDO), into calls on the Script Support Library.

Once done, however, the enhanced client-server features of TP Manager serve to enhance the integration capabilities of the Scripted gateway, and vice versa. For example, suitably written scripts can enable the serial re-use of host sessions by many clients, in common with other TP Manager servers (R6.2). Legacy services can also be readily integrated with native (e.g. RDBMS-based) services.

As well as many-to-one serially re-usable services, TP Manager also offers extended one-to-one conversational associations between clients and services (and in this case host sessions). These can be used to satisfy end-to-end security requirements where necessary (R7.2) or to export low-level host session control back to the client (R4.1).

3.8 Interactive Script Generator

Finally, an interactive terminal emulator has been provided, known as the **isgen** toolkit. This uses the above libraries to access the target host, so it shows exactly the image that scripts and clients will see (R9.4, R9.5). Tools are included to capture interactive dialogues as skeleton scripts, to analyse formatted screens, and to replay existing scripts (R9.1). Assistance for raw XHLLAPI usage is also provided (R9.3).

The screen display component is itself a separate library **isgmon** which can be linked into the applications themselves to enable real-time session monitoring (R9.2). The TP Manager Gateway makes use of this capability.

4 Evaluation

In conclusion, the practical effectiveness of the scripted terminal-emulation technique is considered, as indicated by some initial experience with the resulting product set.

It is not unreasonable to suppose that the gateway described above can appear to clients as a native network service, and to host systems as a native terminal. It is also not unreasonable to assume that a virtual terminal can be appropriately controlled through a scripting language - a number of commercial products demonstrate this capability (though they may often be limited to providing GUI's to single applications). Thus, in principle, any character-based host system and application ought to be accommodated.

The cost-effectiveness of the approach compared with the re-engineering of the host application as a network service is a central issue, and will depend on the circumstances. Re-engineering may just involve modification of some middleware to conform to the host platform's distributed programming architecture, if any. Or it may involve porting to a more suitable platform, or even a complete rewrite.

If the scripted gateway already supports a suitable terminal emulation, its cost as an alternative to re-engineering the host depends chiefly on ease of scripting, which in turn depends on the behaviour of the target host application.

Some general classes of host behaviour can be identified and are discussed below. From the viewpoint of scripting the main distinguishing attributes are:

- synchronisation with host output (how does the script know when results are available on the virtual screen, and when to start typing again?)
- location of fields on the screen (are they fixed? Is there a standard screen layout? Do fields have any distinguishing characteristics? Does the screen scroll?)
- non-deterministic behaviour (are there any unsolicited host outputs such as operator broadcasts? Where do they appear?).

4.1 Buffered protocols

Buffered or block-mode protocols like 7561 and 3270 seem the easiest to handle. They have explicit *turn-control*, so the gateway can inform scripts exactly when host output has finished. Likewise, explicit field delimiters are included on the (virtual) screen so that automated screen-analysis is possible. Finally, unsolicited outputs tend to be constrained to specific broadcast or status lines. (Those applications which operate buffered protocols in scroll-mode, like VME MAC, can benefit from some of the provisions for asynchronous protocols discussed below.)

Buffered protocols, therefore, offer much scope for automation of sessions. The tools developed to date (ISG toolkit) allow interactive dialogues to be captured directly as scripts, with screen contents unpacked automatically into script variables. Scripts can be replayed without modification, but usually some manual adjustments are needed: parameterisation; exception handling; pruning of unwanted fields, etc.

4.2 Asynchronous protocols

Asynchronous protocols, like VT220, generally require scripts to be more intelligent. There is no explicit turn-control, so a script (like a human) has to deduce from screen content changes when host outputs have finished. Likewise, there are no explicit field delimiters, and usually no particular conventions regarding unsolicited messages. Therefore, some specific support is provided by the gateway, according to the particular style of screen operation.

4.3 Scroll-mode operation

Many applications of asynchronous terminals still use the simple teletype scroll-mode. The interactive UNIX Shell is an example, as are public services targeted at lowest common denominator terminals. Again, applications with complex HCIs often provide a command mode for batch operation. This simplicity of presentation leads to simplicity of scripting, provided turn-control and field location can be resolved.

Usually, completion of host output is detected simply by looking for a specific prompt, like the UNIX "$ ". The problem is distinguishing a new prompt from previous prompts, particularly the most recent which may be in the same place that the new one is expected to occupy. To help, the asynchronous terminal emulators keep resetable statistics, e.g. of things like numbers of scroll operations performed. Therefore, a script can defer checking for a prompt until a certain number of scrolls have taken place, meaning that the host has at least begun its response and will have scrolled the last prompt out of the way. Alternatively, though less efficient, the old prompt could be erased from the virtual screen using the screen-update facilities provided for buffered protocols.

A further scripting feature provided for scroll-mode access is the ability to mark the current cursor position on the screen with a flag which then sticks to the screen contents and scrolls with it. Subsequently, the marker and current cursor position will delimit the host dialogue since the marker was placed. Facilities are provided to retrieve screen data based on these delimiters, either as an unstructured sequence of lines, or by using the marker as an alternative addressing base for formatted data which some hosts output in scroll-mode.

4.4 Screen-mode operation

Asynchronous host applications using formatted screens can be simpler to script than scroll-mode as far as field location goes, but deduction of turn-control can be more difficult (unless, of course, they provide an alternative scroll-mode presentation). Drop-down menus and overlaid windows present additional complexity.

Again, statistics can be used, though screen-access optimisation in the host can make things less predictable. The cost is entirely governed by the host application's uniformity or otherwise, and how much of its functionality is to be accessed. At one extreme, using scripting to operate a large, complex application in its entirety, say to provide a GUI, will no doubt prove a major exercise. On the other hand making only a few of its features available to network clients (say the mail and/or diary facilities of an office automation system) may prove extremely cost-effective compared with re-engineering.

The gateway provides some specific assistance for asynchronous screen-mode operation, namely by helping to locate fields in the absence of explicit field delimiters. Thus, changes of character attributes, line-ends, and extended white space are interpreted as field delimiters. This enables the same tools to be used as for buffered protocols - generation of scripts can be partially automated, though more subsequent editing may be needed.

4.5 Performance

Little hard-evidence yet exists regarding the performance of the gateway. Clearly it would not expect to achieve the same response or throughput as would purpose-built servers. However, users have so far compared gateway performance with existing native-terminal access to their legacy systems, and have, therefore, been quite satisfied with the improvement achieved (due to automation, host session sharing, exploitation of client processing power, etc.).

4.6 Future directions

Actual usage of the gateway to date has been very encouraging. In the future, it is expected to develop in a number of ways, e.g.:

- additional terminal emulations
- support for other client-server platforms
- enhanced tools
- enhanced script support (especially declarative languages).

5 Conclusion

To summarise, the scripted gateway approach:

- is cost-effective - integrates legacy programs and systems, without change, into new client-server applications;
- is **transparent** - appears to legacy systems as a native terminal, and to clients as a set of native networked services;

- has **wide host applicability** - can support a variety of terminal emulations (e.g. VT220, 7561, 3270), over various communications media;
- has **wide client applicability** - available as a TUXEDO server, and as libraries to build gateways for other client-server environments;
- is **flexible** - supports various levels of client access to host sessions;
- is **configurable** - replaceable modules include the scripting language (UNIX Shell) and communications routines.

Acknowledgements

UNIX and TUXEDO are registered trademarks of UNIX Systems Laboratories, Inc. in the USA and other countries.

DEC and VT220 are trademarks of Digital Equipment Corporation.

Biography

Paul Duxbury

Paul Duxbury joined ICL in 1973 after graduating in Mathematics from Imperial College. He worked initially as an engineering test programmer in West Gorton, Manchester, before moving to VME development in Kidsgrove. Since 1980 he has worked on the DRS range of products, including lead design of the DRS300 CDOS system and DRSNET. More recently he has worked in a wide variety of fields including communications, TP, systems management and security. He is currently a senior designer in the Operating Systems Extensions Centre, Kidsgrove, which specialises in commercial extensions to UNIX and other server platforms.

The Management of Client-server Systems

Barrie Archer
The SOLUTIONS *Centre*, OPEN*framework* Division, ICL, Bracknell, UK.

Abstract

The deployment of distributed computer systems employing a Client-server Architecture has increased the awareness of the need for distributed systems management solutions to support such systems. This paper analyses the systems management implications of client-server systems, develops models for the organisation of systems management, indicates the types of solutions that are required and outlines possible future directions.

1 Introduction

Historically the capability to manage computer systems has tended to lag behind the functions that needed to be managed. In some ways this is inevitable since it is only as this functionality is deployed in real production situations that the detailed requirements for management become apparent. For example, as computer systems became more powerful it became clear that facilities were needed to enable the various software and hardware components of the system to be controlled so that they might deliver service to end-users. Hence mainframe systems have typically come with a rich set of facilities to enable them to be managed. Mid-range systems have followed with management becoming increasingly important as the power of the systems increases and enterprises rely on them for day-to-day operations.

As enterprises build systems using a client-server architecture, so the requirements for managing such systems become more detailed and solutions specifically oriented to such management become available.

Thus, the importance of systems management in client-server systems is becoming increasingly recognised -

> *"Because of its critical role in the deployment of client/server networks, systems management functionality is an increasingly important component of the functionality that servers must provide [GARTNER-SMEM, 1993]."*

This paper looks at the specific requirements for managing client-server systems. The majority of these requirements are for managing clients, since this is where the greatest gap exists between what is available and what is required. Many requirements for managing the server are the same as for managing Mid-range systems, and currently these are much more adequately covered by the facilities available. There are, however, specific requirements for the server, particularly to support distributed server resources and to provide the server end of client management functions.

The categorisation of systems management tasks used in this paper is based on Gartner's Midrange System and Server Evaluation Model (MSSEM) - see [GARTNER-SMEM, 1993]

2 Client-server overview

2.1 The definition of client-server

For the purposes of this paper, the definition of client-server systems follows that given by Gartner and adopted by ICL's Client-Server Systems, namely:

> "*Client-Server is the splitting of an application into tasks that are performed on separate computers, one of which is a programmable workstation (e.g. a PC).*"

In practice, it is found that many client-server systems will not consist of a homogeneous mixture of systems. In particular, services will be provided in the most cost-effective way with interoperability enabling these services to be provided on different platforms. The realisation of this heterogeneity will be different in different circumstances, sometimes applying to hardware, sometimes to operating systems and sometimes to applications (e.g. databases) and, all too often, to all three. This has particular significance for the management of client-server systems.

2.2 Types of workstation

It is important (from a systems management view) to distinguish between workstations (clients) used as personal productivity tools and those used to distribute processing power, since the management requirements are normally quite different. We will call these two types *personal workstations* and *application workstations* respectively.

In figure 1 we summarise the differences between these two types of workstation, categorised by 6 attributes (described in column 4 of the table).

This assignment of attributes to the two types of workstation is not intended to imply that all workstation instances will fall definitively into one class rather than the other. There will be examples of workstations that are mainly application workstations with some personal workstation functions, and vice versa. Nevertheless, understanding that there are these differences is of benefit when considering the type of management required for a client. Later we will refer to this classification in describing management tasks and their applicability.

	Personal workstation	Application workstation	Description of Attribute
configuration	diverse	standard	the hardware of the workstation and the software that is run on it
data	local	remote	the location of data: local to the workstation or remote, that is on the server
applications	personal	enterprise	the type of application: personal productivity or enterprise defined
administration	local	remote	how administration tasks are performed: locally on the workstation or remotely from the server
security	complex	simple	whether a simple logon gives access to all data or whether security varies between resources
networking	peer-to-peer and client-server	client-server	whether limited to interaction with a server, or peer-to-peer networking with other workstations

Figure 1 Attributes of workstation types

2.3 Management of client-server systems

In managing client-server systems, it is important to appreciate the significant difference between the requirements for managing the server and those for managing the client. Generally, we can say that for the server the requirement is for a small number of systems to be closely monitored and accurately tuned. For the client, the requirement is for a large number of systems to be managed as a whole, with relatively loose monitoring and accurate and detailed inventory control of hardware and software.

On the basis of these requirements, the management tasks, and hence the tools needed, have significantly different characteristics between the client and the server. This is not to say that the tools themselves, or the interfaces to them, have to be different but it does mean that tools should be selected at least as much for suitability to the management tasks as for commonality between client and server management. Take as an example performance monitoring. It may be preferable to choose one tool to perform the relatively simple monitoring on a workstation but another tool to perform the complex monitoring, and capacity planning, required for the servers and network.

At this point it is worth commenting on the possibility of having server and client machines as the same physical hardware. Although such an arrangement may be possible, and may apparently offer cost benefits, it should be avoided since, compared with clients, servers have very different requirements for quality of service. From a management point of view, many difficulties can arise in this situation principally because the management responsibility is no longer clear cut. In the end it will be found that the increased on-going costs will outweigh the initial savings.

3 The cost of management

The use of clients in a client-server network has often been the domain of computer-literate professionals using personal workstations. The effect of this has tended to mask the costs of administering the workstation because each user has performed the necessary administrative tasks in the course of using the system. The fact that the time spent administering workstations is not separately accounted for has tended to favour them in any cost analysis. Several factors have arisen which are making this cost more visible.

- Workstations are being used by a wider range of people, increasingly clerical staff with only minimum necessary computer literacy (often using application workstations).
- Workstations themselves are becoming more complex to administer, particularly their communications with the network and the servers on it.
- Professionals are becoming less willing to spend the time necessary to administer their personal computers.
- Management has become aware that personal workstations are often a vital corporate resource, but the administrative tasks necessary to secure this resource are not always being performed.

3.1 Evaluating costs

The true costs of management can be considered as the costs of *doing it* compared with the costs of *not doing it.*

The costs of management may be broken down into personnel costs (administrators and their management, as well as user costs as described above) and resource costs, that is the cost of machines to run management applications, software to perform management tasks plus an incremental cost for each resource managed. This last item includes any additional hardware required to make a resource manageable (e.g. extra store) plus a software cost where, as is becoming more common, management application licence fees are based on the number of items being managed.

The costs of not doing management can often be hidden, for example inadequate service, user/customer dissatisfaction, risk taking, inefficient use of resources, inadequate accounting, etc. We have pointed out that the users are often performing administrative tasks without the cost being accounted for. However, there are also the hidden costs due to users making catastrophic errors, failing to perform necessary management tasks on an adequately regular basis or using resources inefficiently. The cost of risk taking is normally hidden until the risk turns into a reality, so, for example, the cost of not performing a backup of workstation data only occurs when a hardware fault requires large amounts of work to be done again.

The hidden costs of delivering an inadequate service to users result in a lowering of productivity; this cost may be better understood through the implementation of Service Level Agreements. These lay down specifications for the quality of service that will be supplied to the user and many organisations are now implementing such agreements. Systems management is particularly useful in monitoring adherence to such agreements and in providing a means to predict potential future shortfalls, so enabling proactive measures to be taken.

3.2 Reducing Costs

One specific aim of systems management is to reduce costs but this should be taken to include the hidden costs identified in the previous section. Hence, it will only be where hidden costs are fully appreciated that the true gains of implementing systems management will be recognised.

Some of the ways in which systems management addresses direct cost reduction are:

- automation of mundane tasks
- efficient utilisation of expert resource
- elimination of human error
- scalability to large numbers of components
- efficient and accurate information collection

All of these areas apply to the management of client-server systems and some take on particular significance because of the factors outlined before.

The areas of indirect cost reduction addressed are concerned with performance monitoring, capacity planning, change management, data integrity, etc. Some of these areas may not be applicable to client systems, particularly for personal workstations, although benefit can still be gained by ensuring that user responsibilities are fully understood both by individuals and management. For example, if a user is responsible for performing backups of personal workstation data, this should be clearly understood.

3.3 Change management

One of the advantages of client-server systems is that they can easily be changed to accommodate increased workloads or new applications. A new workstation can be installed or new software can be used at a strictly incremental cost. However, the disadvantage is that there is a certain anarchy which works well enough when there are no difficulties, but which gives rise to a heavy management task load if problems arise or there are knock-on effects of changes.

Systems management has a role to play both in the implementation of change and in the verification of change. By implementing change in a systematic and structured fashion, the chances of unexpected and unwelcome side effects can be greatly reduced. Keeping accurate and up-to-date records greatly simplifies the planning of change, and even, in the extreme, eliminates the need for change altogether. For example, *shelfware*, hardware and software that is purchased but never deployed, can be eliminated or used to meet new requirements.

4 Management model

In this section we explain how the management of systems can be represented by different models which help us to understand the relationships between different styles and classes of management.

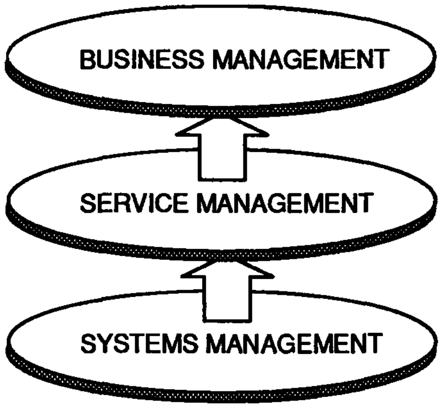


Figure 2 Levels of management

4.1 Levels of Management

Systems management can be viewed as the first level of a hierarchy of management disciplines, the higher levels being *service management* and *business management*, see figure 2. This is explained in detail in

[OPEN*framework*, 1992]. The basis of this concept is that the information obtained and the control exercised by systems management, can be abstracted to enable comparable tasks to be performed at both service and business levels. At the service level the concern is about how the Information Technology (IT) services on which an enterprise depends are functioning (e.g. the assurance that an order entry system is functioning correctly). At the business level the concern is about the overall impact that the IT services can have on the business of the enterprise (e.g. the contribution of the order entry system to the business).

However, in this paper we are concerned with examining systems management with particular reference to client-server systems. The concepts of service and business management fit on top of client-server systems management without regard to any peculiarities or differences in the client-server environment. Indeed, it is fundamental to the concepts of these higher levels that they are independent of specific systems management functionality.

The rest of this paper will consider client-server systems management without regard to service and business management. Please consult [OPEN*framework*, 1992] for further information on this subject.

4.2 Centralised management

From the point of view of corporate departments, such as Information Technology, Management Information and Finance, centralising the management of computer systems has many benefits in terms of controlling computer resources and the amount of money spent on them. A consequence of this approach is that there is often a degree of standardisation in the configuration of clients, both in terms of hardware and software.

In figure 3 we illustrate two ways in which central management of workstations can be effected. If the resources of the workstation are exported to the server, then they can be managed using the server administration facilities either as a side effect of the management of the server resources or specifically as workstation resources that are resident on the server. Alternatively, the management of the workstation can be effected by remote administration from the server.

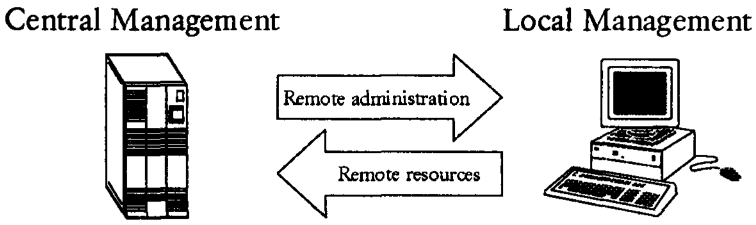


Figure 3 Central and local management

Remote administration is more often employed for personal workstations, whereas management through remote resources more often occurs on application workstations.

4.3 Devolved management model

One of the effects, and in some cases the motivation, for implementing client-server solutions is that it gives power to the end-user, or end-user departments, to define the way in which Information Technology will be used to provide solutions to end-user-tasks. We say that this is *devolved responsibility*. End-users are given responsibility to perform a task and authority to define how this should be achieved. So, for example, requirements for processing a particular class of company data may be defined at a corporate level but the end-user department may be free to choose how this should be accomplished.

This devolved responsibility obviously has to be held in balance with the centralised management described in the previous section, and we are particularly concerned with the management implications of this. We see that for client-server systems there needs to be a clear understanding of how responsibility for the management of clients should be divided between the end-user and the systems administrators.

Obviously this particularly applies to personal workstations, running a range of applications some of which may have definite corporate implications (e.g. mail agent software which provides a user interface to a mail server) and some of which are more user task oriented (e.g. personal productivity tools, such as spreadsheets). However, we will develop a model into which the application workstation also fits.

In order to ensure that the client can be managed it is necessary to define:

- what needs to be managed
- who is responsible for the management
- how the management will be effected.

We illustrate this diagrammatically in figure 4. We show responsibility for the management of the workstation split between the user and the central administrator. For this model to work there must be a clear definition of the management tasks that the user is required to perform. (The user may choose to perform additional management tasks in exercising the responsibility that has been devolved, but that is not of concern to this model.)

The area of overlap may be realised in two ways

- the user performs a task and the central administrator checks (preferably automatically) that the task has been performed, examples being local back-up and changing passwords.

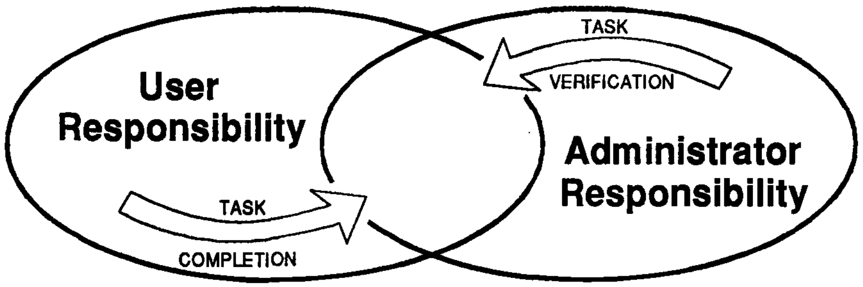


Figure 4 Allocation of management responsibility

- the central administrator performs part of the task, which is completed by the user, an example being software distribution when the user is responsible for the final installation phase.

5 Management tasks

In this section we look at various tasks to be performed to manage client-server systems, highlighting areas that are particularly important or different. As stated before, we are primarily concerned with the management of clients since the management of servers is generally the same as for mid-range systems.

5.1 Operations management

5.1.1 Accounting and chargeback

For clients, determination of resource usage (accounting) and allocation of charges to end-users or end-user departments (chargeback) is normally limited to costs for the client hardware and the software installed on it. Hence this area is closely related to inventory and licence management. It tends to be more complex for personal workstations than application workstations because the latter tend to have more standardised hardware configurations and fewer, more closely administered, applications.

5.1.2 Network management

The impact of network management on the client is limited to managing it as a component of the network. Hence it is important that the workstation software network components can be managed remotely, implying the capability to act as an agent to a Network Management Station (NMS). So, for example, the workstation should respond to NMS requests to identify itself, thus enabling the NMS to build a topographical map of the network.

5.1.3 Print management

The requirements for print management depend very much on the way in which a client-server system is organised. For client-server systems that are strongly workgroup-oriented a simple print management capability may be sufficient, handling only printers within the workgroup.

However, where the client-server system is (also) serving enterprise needs, a sophisticated, all-pervading print management system is likely to be required. This would have the capabilities such as:

- ability to choose appropriate printers according to attribute
- allocation of print jobs dynamically
- provision of sophisticated queue manipulation
- handling exceptions
- providing statistics
- resilience to failures
- security features

For a personal workstation much of this functionality will be hidden, although manipulation of queues will be visible. For application workstations even this visibility may not exist.

5.1.4 Problem handling

This is concerned with determining that an operational problem has arisen and progressing its resolution and correction. Both manual input (support calls) and automatic detection of faults are relevant to client-server systems. The former are likely to be in the majority for clients although automatic detection is becoming increasingly available and offers the promise of much more efficient administration.

The first aspect of automatic detection is the monitoring the resources being managed. This covers both the handling of exceptions and faults, and the regular inspection of the resources in order to recognise conditions that indicate a problem may occur in the future. A critical part of this is filtering to ensure that administrative staff are not swamped by minor or self-correcting faults. This is particularly critical for client-server systems where some faults will be dealt with locally by the user.

Once a problem has been detected, there must be the capability to diagnose the fault (testing, remote access, etc.) and to track the work being done to diagnose and correct the problem (using a help desk plus tools specific to the problem, such as software distribution or configuration).

5.1.5 Application monitoring

Application monitoring is restricted mainly to servers and application workstations. The software on the workstation is effectively a distributed part of the application running on the server and hence needs to be considered in presenting a total picture of the health of the distributed application. For the workstation, an extra degree of flexibility is required to take into account the unpredictable work pattern of any particular user. For example, it is necessary to take into account the workstation being switched off when the user goes on leave.

5.2 Performance management

5.2.1 Capacity management

This involves measuring the resource utilisation of various components of a client-server system and actions taken as a result of that determination. This includes charging for resource usage (accounting), planning for future capacity requirements and possibly providing data for service level management (see 5.2.2).

This type of function tends to apply more to servers (and the communications between clients and servers) and to application workstations (typically concerning response times). Users of personal workstations will often be left to determine from experience when there is a capacity problem. This method is not very efficient but, equally, the impact on the enterprise is also much less.

5.2.2 Service level management

Devolved responsibility often results in the responsibility for the servers lying with different parts of an enterprise from the clients. This split is likely to increase awareness of the need to define the level of service to be provided by servers, encapsulated in Service Level Agreements (SLA).

Hence, we see that client-server systems may lead to an expansion in the number of agreements and therefore to applications to manage these. This will involve determining the level of service offered and comparing this with what is specified in the agreement.

Although much of the functionality needed to manage SLAs will be server-based, the requirement can be seen as an aspect of managing client-server systems.

5.2.3 Performance monitoring

Most personal workstations would not be monitored for performance. Application workstations are more likely to be monitored in order to obtain information about quality attributes perceived by the end-user, for example, response times. Facilities to provide this monitoring are likely to be provided within the application running on the client.

5.3 Storage management

5.3.1 Archive

Archiving involves the removal of data from immediate storage to off-line storage. There are often requirements for longevity of the medium on which it is stored as well as for large quantities of data. Archiving is infrequently required for workstations but, if needed, can be provided via a server.

5.3.2 Backup/Restore

Backup of directly addressable data ensures that, if the data is lost, it can be restored to its state at a previous point in time. There are basically the following options for backup of client data:

- Where the data is held remotely on the server, backup of workstation data will occur when the backup of the server data occurs. The problem with this solution is that it does not allow the end-user to restore the data.
- Backup of the workstation data can be performed by the end-user using a server as the repository for the backup. The advantage of this is that the administration of the server is likely to include off-site storage of the backup data.
- Backup of the workstation data to local removable media can be performed by the end-user. The disadvantage of this is that it can be difficult to verify that it has been done and the media stored correctly.
- Administrative action from the server can backup the data on the workstation. This requires appropriate facilities for the server to access the workstation and for the workstation to be available for the operation (probably during unsocial hours).

Backup of the server data, essential for some of the above to work, is covered by the normal administrative activities for servers.

It is important to appreciate that data on a personal workstation can easily be of vital importance to an enterprise (e.g. the result of an extensive company audit could reside on one workstation), and yet adequately securing it is often overlooked.

5.3.3 Threshold monitoring

Monitoring resource thresholds on a workstation is needed to ensure that it does not cease to function through lack of resource. The most common such resource is filestore. Depending on the use made of the workstation, it may be necessary to monitor with only a very coarse granularity, for example daily.

5.4 Security management

5.4.1 Authentication

Authentication assures that the user has the identity claimed. Authentication at the workstation can become especially important when the workstation is trusted by the server, that is where the user performs one logon which provides authentication for all the servers to be accessed and all the functions to be performed on those servers. Application workstations, where the authentication is likely to involve the server, are easier to make secure. Making a personal workstation secure is likely to involve a restriction on the user's capability to manage it.

5.4.2 Intrusion detection

Detection of viruses is particularly important, especially with personal workstations. Various mechanisms are now available for virus checking and these can be initiated centrally or locally. It may also be necessary to institute restrictions, using standard access mechanisms, to inhibit the spread of viruses through the network.

5.4.3 Audit
Audit means keeping a record of the identity of the user who performed a particular action and, hence, is only as good as the authentication of the user. Secure audit is more applicable to an application workstation but may also be implemented on a personal workstation.

5.4.4 Password management
For application workstations, passwords are likely to be managed from the server. For personal workstations, password management is likely to be limited to verification that the workstation has password protection for resources deemed to need it and that the passwords are secure (e.g. are not too simple and are changed regularly).

5.5 Configuration and change management
5.5.1 Asset management
One consequence of devolved responsibility is that it may be difficult for the enterprise to ascertain what hardware and software it actually has. Some enterprises might decide that end-user departments should have total control of IT hardware and software but most will want to have central asset management for the following reasons:

- accounting
- planning
- support

Asset management involves determining the configuration (hardware and software) of a workstation dynamically and providing a means to store this information and access it in a flexible manner, so that, for example, reports can be generated. Whilst this also applies to servers, these would normally be under closer supervision, enabling records to be more easily kept. However, the asset management system should handle both clients and servers.

5.5.2 Licence management
Enterprises are becoming more aware of the need to ensure that only properly licensed software is used on both clients and servers. Determining who is using particular licences may involve positive control (inhibiting use of software without a licence) or passive control (auditing). Floating or network licences allow licences to be shared between users by providing a network-wide repository of licences that can be allocated to users on demand.

Portable (laptop) computers present a particular problem because vendors often permit a single licence to be used on two (or more) machines provided only one person uses the software. Future development like payment for usage will increase the management facilities needed in this area.

It is important for client-server systems that the licence management can be performed by an administrator in a distributed manner. Hence, technologies that require interactive dialogues on the workstation (challenge systems) will not scale well to large numbers of clients.

5.5.3 Configuration management

Configuration can apply to both hardware and software. Today's workstations, particularly personal workstations, are highly configurable, and this can make central management difficult. For this reason, in some management regimes, there may be restrictions on the permitted variability of configurations. For the effective central management of large numbers of workstations, non-interactive configuration is called for.

5.5.4 Software Distribution

In client-server systems, software distribution should normally be done electronically, except where the software is for a single system or where the connection is unsuitable against this for reasons of speed or cost. Even in this latter case, software may be distributed on a removable medium but management control effected centrally. Facilities need to be provided to enable software to be distributed from one or more distribution points with the capability to perform the installation remotely. There are many different detailed requirements depending on the systems and users.

Server software would normally be under strict control and its distribution, installation, etc., would be handled by administrators able to manage all the servers in a system.

For workstations, software may be controlled entirely by administrators or may be controlled by the user, or a combination. The choice depends on the software and the user. For example, some organisations may insist on central control of operating system software but be happy for users to manage their own applications. What is required by administrators is a consistent way to administer software on all systems, irrespective of the hardware, software, communications, etc. Such administration would include facilities such as:

- distribution of software
- installation of software
- removal of software
- configuration of software

For managing large networks of clients, installation processes that require an interactive dialogue with the person doing the installation are not practicable.

6 Technologies available

Ideally the tasks defined in section 5 should be implemented in a way that allows for integration between them, so that when performing one task the information or control facilities of the other tasks would also be available. So, for example, problem handling could use information in asset control, or service level management could use the information gathered by capacity management. This integration could be achieved by having one management application that implemented all the tasks or several such applications that conformed to a common interchange standard enabling them to interoperate.

At the present time there are, in general, management applications that address each of these tasks in depth, or management applications that address them all in somewhat less depth. Interoperability between different management applications, which depends on there being agreed standards, is still at a very basic level. Some management applications will interoperate very successfully with specific other management applications, but only on a one-off basis.

Currently, the only management infrastructure commonly used and deployed in client-server systems is the Simple Network Management Protocol (SNMP). Originally developed to manage network components, this protocol can be used to enable some management tasks to be remotely performed on workstations (particularly monitoring and configuration).

7 Future directions

Standards for managing workstations have been defined by the Desktop Management Task Force [DTMF, 1993], an industry consortium of companies concerned with desktop systems. These standards define interfaces for the transmission of information between management software and manageable components on the workstation and for how the manageability of the components is described. These standards are likely to be important in providing a means by which workstations become more manageable by central administrators.

The Distributed Support Information Standards group [DSIS, 1993], an industry consortium of companies concerned with servicing, has been defining the components and attributes of those components that need to be manageable in a distributed manner. Conformance to these standards by vendors of hardware and software will result in a homogeneous management view of client-server systems even when composed of heterogeneous components.

Current initiatives, particularly those of the Object Management Group (an organisation set up to promote standards in distributed object oriented technologies, [OMG, 1990]), are aimed at the distributed use of object oriented techniques, enabling applications to make use of objects distributed throughout a network of systems. Such techniques are likely to be of major importance in the construction of client-server systems.

8 Conclusion

The management of client-server systems imposes particular requirements on the way an enterprise organises its systems management and on the tools needed, by both administrators and end-users. However, management of clients is less advanced than that for servers, particularly in terms of tools to manage populations of clients.

We have shown that the type of management depends on the type of workstation and that there needs to be a clear understanding of how responsibilities for systems management are split between end-users and central administrators.

Proper attention to these aspects of client-server systems can effect real financial savings, increase user satisfaction and productivity and significantly reduce the risks to the enterprise from malicious and accidental disasters.

References

GARTNER-SMEM, 1993 Gartner Group, Midrange Computing Strategies, Key Issues K-290-1414 P. McGuckin, The Gartner Group Inc, Stamford, CT, USA. December 8, 1993.

OPEN*framework*, OPEN*framework* Systems Management Architecture, Prentice Hall, 1992.

DMTF, 1993 Desktop Management Interface Specification DRAFT 4.2 The DMTF can be contacted on +1 503 221 2945. January 12, 1994.

DSIS, Distributed Systems Support Information Standards Requirements Specification, Doc PW017, Rev 1.0 (11 June 1993) obtainable from Ray Edgerton, DSIS Group Chairman, BABSS, 50 East Swedesford Road, PO Box 3004, Frazer, PA 19355-0704. Phone +1 215 296 2159; also by anonymous FTP from gatekeeper.dec.com: /pub/forums/dsis.

OMG, Object Management Architecture Guide, 1.0, The Object Management Group, Inc., Framlington, MA, USA. November 1, 1990.

Biography

Barrie Archer

Barrie Archer is a Systems Designer working in *The* SOLUTIONS *Centre* of ICL OPEN*framework* Division. He is responsible for Systems Management aspects of integrating ICL's client-server solutions. He has worked on the strategy of ICL's Systems Management products and participated in the development of ICL's OPEN*framework* Architecture for Systems Management. He is the ICL representative on the X/Open Systems Management Working Group and POSIX 1003.7 Working Group. Electronic mail address 'barcher@oasis.icl.co.uk'.

Dialogue Manager: Integrating disparate services in client-server environments

Roger Thompson and Ian Robertson
OPEN*framework* Division, ICL, Bracknell, UK

Abstract

Many organisations have disparate IT services, using a variety of technologies, which are central to their business. Today's climate often implies rapid change in the nature of that business, particularly in competitive levels of customer service. The current services remain valuable assets, but their full potential cannot be realised while they remain isolated. Re-engineering the services is ruled out for reasons of timescale, cost and risk to the business.

Dialogue Manager is a set of software technologies which resolves this dilemma, by allowing systematic integration of such services at the desktop without changing them, so they can be viewed as a single coherent system, accessed via a single consistent user interface. Existing services can remain fully operational alongside the integrated service, allowing controlled and phased deployment, thereby eliminating business risk. The Dialogue Manager client architecture provides for modular separation of components, with the ongoing ability to adopt the best and most appropriate tools from the software industry.

1 Introduction

"*No one can plan the future. Three years is long-term...Five years is laughable*" [McCracken in Prokesch, 1993].

The 1990s were predicted to be a turbulent time in business. That prediction has been amply fulfilled. While the nature of many businesses is changing at a bewildering rate, the IT systems on which they depend have often not kept pace.

[Brenner, 1994] illustrates some of the benefits of Client-server systems in addressing these problems. Such systems are flexible enough to allow components to be changed, or recombined in different ways, many of which could not have been predicted, and yet maintain the integrity that the business depends upon.

While new systems can be constructed in this fashion, the IT legacy with which most companies currently find themselves is far from an ideal starting point from which to base a client-server infrastructure. However, the time, risks and costs associated with populating that infrastructure from scratch, or by re-engineering legacy systems, may be daunting.

Issue	**Characteristics of solution**
In order to maintain competitiveness, transactions from disparate services need to be combined together. These services were typically not designed to work together	Being able to construct apparently new services from a combination of existing services, perhaps supplemented by new services, provides opportunities to gain competitive advantage. Because these services are large, any solution must be scaleable, implying the use of mechanical tooling for development and maintenance.
Services often have different user interfaces, each of which requires extensive training. This causes both high training costs and poor staff mobility, motivation and productivity. Attempts at service integration without addressing this problem have usually proved unsuccessful. Many services are becoming 'customer-facing'. Current user interfaces are not acceptable in this environment.	The services need to be viewed as if they were a single integrated service, accessed via a consistent user interface. This interface should not be constrained by existing user interface characteristics. The solution should not be tied to a particular toolset, but allow choice from the best that the IT industry can offer.
The business climate requires IT systems be changed as fast as the business demands. Often, the rate of business change is constrained by the rate at which IT systems can change.	Change to IT systems must be incremental. Large scale projects should be divisible into smaller increments, each of which shows rapid return on investment. Later phases should be capable of adapting to changes both in the business and in technology. Conversely, small scale projects must demonstrate that they do not lead to dead-ends.
Businesses are critically dependent on their current IT systems	Improvements or re-engineering must be developed and deployed without disruption to existing systems.

The IT legacy is a valuable asset, but how can it be made to fulfil its potential in client-server systems?

In trying to answer that question, it quickly becomes apparent that some issues are widespread, and particularly pressing. These are summarised in the table on the previous page.

Dialogue Manager was developed to address these issues, providing a client environment in which services can be integrated together systematically and maintainably.

2 Architectural Principles

Dialogue Manager uses the Distributed Function model of splitting client and server [Gartner, 1992]. This model identifies two client components as application logic and presentation. Dialogue Manager recognises the functional separation between these components. It also recognises that the interface between client and server should be described at as high a level as possible, in order to insulate the effects of changes of one half on the other.

2.1 Data definitions - an introduction

Dialogue Manager allows the appropriate data to be viewed in an appropriate way by each client component. This is achieved by encapsulating the data that passes between components. This involves:

- normalising the data
- controlling data relationships by a data definition language
- providing an event-driven to block mode mapping.

The rest of this section examines each of these topics in turn.

Figure 1 illustrates the purpose of data normalisation. On the left of the diagram, different device classes have different interfaces. The choice of Human Computer Interface (HCI) toolkit may be influenced by the number of device classes that can be driven, or the number of different platforms supported. However, it should not constrain how the application logic is to be written, nor what services can be accessed. The alternative presentation logics shown illustrate that, however versatile the HCI toolkit, different users may have fundamentally different requirements on how they use the system, reflected not only in look and feel, but also in dialogue flow. Nevertheless, these differences are not relevant to application logic, nor to the services accessed.

On the right hand side of the diagram, different servers have different networking requirements, and different data interchange formats and rules. These should not be of concern to the application logic, nor to presentation logic. By defining normalised data at the boundaries shown, the necessary component independence can be achieved.

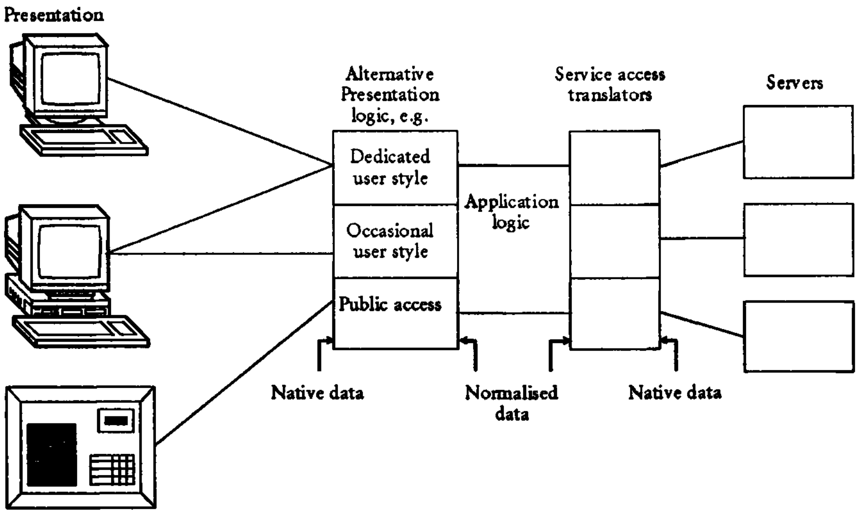


Figure 1 Normalisation of data

A data definition language is used to define logical data sets and relationships between them. To illustrate what this means, consider the example of a mainframe service accessed via dumb terminals. Here the sets of data exchanged between service and terminal correspond to particular screen types, and are determined by the mainframe application. If the same service were designed to be accessed only by program, the data sets might look rather different. The reason for this is that the screen types are constrained by the amount of information displayed on a screen, and by any contextual information required to make sense of it. This may involve constant background text, but commonly also involves retransmission of data from previous screens.

In order to isolate the logical definition of a service from the way it is provided in practice, a data definition language is used. As illustrated in figure 2, this language allows mapping of items between data sets.

Two types of relationship are supported:

- identity, where items have identical values.
- shadow, where the items are related, but require code to transform one into the other.

It is thus possible to construct data sets for new logical services from elements of existing services without disturbing either the existing logical or actual services.

While the mapping could, of course, be accomplished by code, this would be far less amenable to change than data mapping, and far less amenable to mechanical tooling.

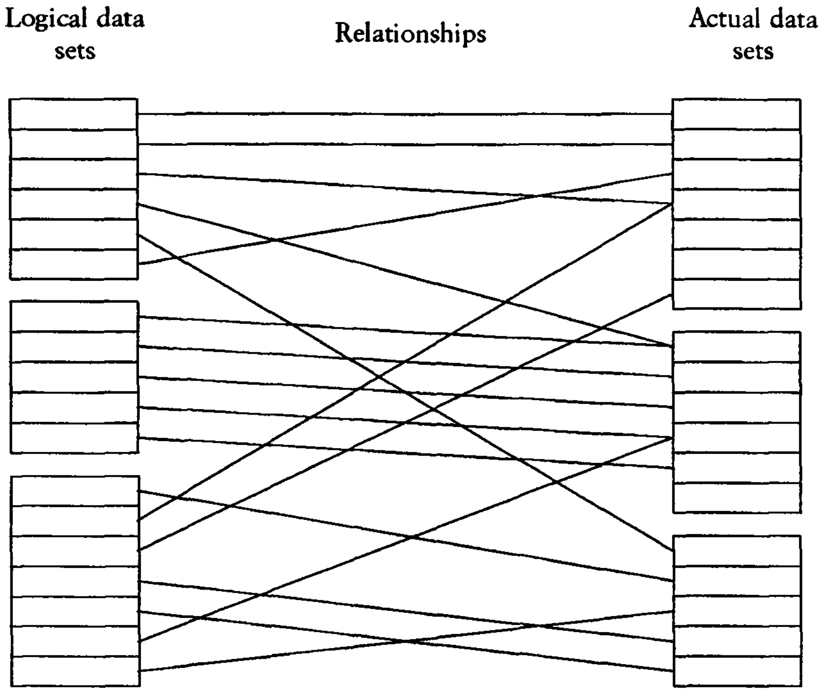


Figure 2 Mapping of items between data sets

The data mapping techniques described above provide data normalisation between application logic and actual services. They are also used to provide data normalisation between application logic and presentation logic. As described above, alternative presentation logic may be required to be connected to the same application logic. The dialogue flow in a particular style of presentation logic should be of concern to application logic only to the extent that it conforms to required business rules. Conversely, presentation logic should not be concerned with what happens to the data it collects, or how the data it presents is generated. Neither component can, therefore, necessarily call on the other to do specific actions. Instead, each component merely triggers the other, effectively saying, "I have changed this data, or I need this data. Do whatever you do in response to this". The data definition language defines which data causes such triggers in which components, rather than by code, for exactly the same reasons as in the data mapping case.

Dialogue Manager provides triggers as events on individual data items. While this is a natural mode of operation for presentation services, external services often deal with data at data set level (screen or block-mode). Interfaces are, therefore, also supplied at this level, e.g. distribute or gather this data set, check validity of data set etc.

2.2 Use of data definitions

In the previous section, data definitions were introduced as a way of insulating components from each other, particularly in regard to resilience to change.

This section describes other advantages that derive from the use of data definitions:

- separation of toolsets, methodologies and skills.
- mechanical tooling to derive and maintain data definitions, and generate run-time structures from them.

Dialogue Manager maps the run-time structures generated from data definitions to a variety of language interfaces. Different components can therefore be written in different languages. It also supports relationships between data definitions in different applications, so that a client can be constructed as a set of separate applications in different languages.

This has several advantages. Apart from allowing toolsets to be mixed and matched from the best that the industry has to offer, the right skills can be applied to the right component. Designers of the presentation component, for example, can be human factors specialists, without having also to be specialists in business processes. The rate of change of technology applicable to the presentation component is particularly rapid. By separating this from application logic, which is rather more stable, the maximum amount of component re-use can be achieved without sacrificing the ability to adopt the best technology available at any time.

The architecture does not demand that separate application and presentation logic be provided. Where the prime reason for using Dialogue Manager is for re-presentation of services, the amount of application logic may be very small - almost approaching the Remote Presentation model of Client-Server. Nevertheless, the use of data definitions ensures that separation can be achieved later without disruption of what has been developed already.

Legacy services can involve large numbers of screens. While it is feasible to derive screen data sets manually for a few screens, this approach becomes totally infeasible where hundreds, or even thousands of screens are involved. In such cases, mechanical generation and maintenance is required. Dialogue Manager has tools for generating and maintaining data definitions from VME™ TPMS systems and TUXEDO® systems. It also has tools for automatic generation of screens for VME TPMS systems, together with the associated processing code. Section 3.4 describes this in more detail. The Dialogue Manager design facilitates the addition of such tooling for other legacy services via the Professional Extension (see Section 3.2).

2.3 Objects

As described in the previous sections, the data description language defines data sets, with relationships and event triggers. The items in a data set have some reason to belong together, for example, they may define a screen or a logical transaction type. The items may be passive data containers, whose significance is known only to the writer of the particular component. Alternatively, sets of items may have active behaviour, provided by code associated with them, in other words they may be objects.

By addressing objects in the same way as other data sets, Dialogue Manager ensures that they can be accessed from a wide variety of languages, and that data relationships can be established between items in different objects, between items within objects and items outside objects, or between objects as a whole.

3 Dialogue Manager features

3.1 Introduction

This section outlines the features provided by Dialogue Manager both at design time and run-time. Figure 3 shows the main Dialogue manager components. These components are explained in sections 3.1 and 3.2.

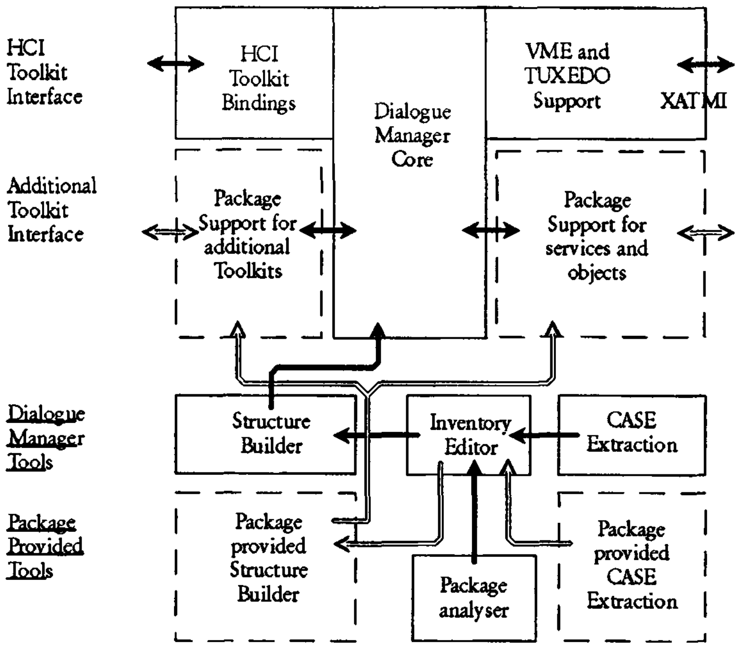


Figure 3 Dialogue Manager components

Design time activities start with running Computer Aided Software Engineering (CASE) extraction tools which can extract data definitions for services mechanically, and converting them into the data definition language used by Dialogue Manager held in an inventory file (Dialogue Manager's terminology for a data set is an inventory). This file is then edited as required, adding relationships, event triggers etc. Runtime structures are then generated. These include structures for:

- language bindings to particular HCI toolkits. These need runtime structures to relate the application's view of data in the particular language to the underlying generic representation.
- structures which encode the generic representation, including relationships and event triggers.
- service access translation structures. These identify particular screen or block types, and define prescriptions for gathering or distribution of data.
- Objects (see section 3.2).

The components for the basic Dialogue Manager product at release 2 are shown in solid boxes in the diagram above. This product allows access to external services via its use of the X/Open supported XATMI interface, a subset of the TUXEDO program interface, the ATMI. Dialogue Manager provides applications with higher level interfaces to external services, which map to XATMI using the support routines illustrated in the diagram. XATMI effectively provides the means to pass data between a client and a server, without the need for the client to handle routing or networking itself. As well as native TUXEDO services, other service types can be accessed via TUXEDO gateway servers, for example ICL VME TPMS (available as part of the ICL TP Manager product), and IBM CICS. XATMI is transparent to data. For example, screen data passing between the client and legacy services is delivered without knowledge as to its contents. Understanding the contents of data passing between the client and legacy services is one of the main keys to front-ending and integrating such services.

As explained above, Dialogue Manager's structures are specifically designed to allow this process to be automated. In the basic product, VME data dictionary definitions and TUXEDO View File definitions can be captured (represented in the diagram above as CASE extraction), and structures can be generated in Dialogue Manager's data definition language in an inventory file. Structures from several services can be generated in this way.

The HCI toolkits supported by the basic product are Microsoft® Visual Basic and Visual C++, and Gupta SQL Windows.

Business logic language bindings are additionally provided for C, C++, and Micro Focus COBOL.

The Dialogue Manager design allows for the addition of other language bindings for both business logic and HCI toolkits (see the next section).

3.2 Dialogue Manager Professional Extension

While the Dialogue Manager standard product offers external service access via XATMI, its core capabilities have much wider applicability.

As illustrated in figure 3, Dialogue Manager also accommodates third party packages. The standard Dialogue Manager Release 3 product will be able to use such packages. However, the development required to allow packages to integrate with Dialogue Manager requires use of the Dialogue Manager Professional Extension.

The third party package may, for example, provide additional HCI toolkits, service access capabilities or object libraries. Dialogue Manager's ability to normalise and relate data items allows packages to be used for integrating services for which they may not have been explicitly designed, and widens their applicability by complementing their specific strengths with those offered by others, or with the facilities provided by the standard product.

Two packages are to be provided initially by ICL for Version 3. The first package allows direct connection to ICL VME services without the use of TUXEDO. The second package allows direct connection to IBM CICS services accessed by 3270 emulation.

Direct connection to VME is provided so that the decision as to whether to access VME via the TUXEDO route can be taken as the business requires it, but also to allow a low entry cost for evaluation, pilots and initial deployments. Applications using the direct connection route are "TUXEDO ready", in that the same package interfaces can also be used via the TUXEDO route.

Direct connection to VME also allows non-TPMS services to be accessed, for example, MAC services. Dialogue Manager provides a 7561 presentation emulator in Visual Basic, so that raw 7561 presentation protocol can be translated into a screen and attribute map, and be displayed if required, with the look and feel of a 7561 emulator screen. These facilities may also be useful for TPMS services where individual fields contain 7561 protocol information (typically for bulletin boards or dynamically configurable screens).

3.3 CASE integration facilities

As illustrated in figure 3, Dialogue Manager has been designed so that CASE integration tools can be used to extract and maintain service definitions. To give examples of such tools, this section and the next describe what is provided with the standard product.

Dialogue Manager provides tools for automatic generation of data definition language files from ICL VME ISDA or Visionmaster™ screen definitions held in DDS (VME Data Dictionary System). Data relationships between identically named and identically defined fields are generated automatically.

Intelligent compression, under user control, is provided to generate inventory types by merging the data definitions for closely related screens. This is an attempt to assist in the process of deducing logical transaction types from the screens that actually exist.

Dialogue Manager also provides tools for automatic generation of data definition language from TUXEDO View files, both for Views and FML Views.

Data relationships are defined in the inventory file. Though some relationships may be generated automatically, many will not be. It is therefore important to be able to maintain such relationships in the event of service definition changes. For this reason, intelligent merge tools are provided.

3.4 Automatic screen generation

Many modern HCI toolkits offer a highly productive environment in which to construct user interfaces. Nevertheless, the work required to improve the HCI for an entire service, or to provide an integrated HCI for several such services may still be substantial. It may be a useful first step to be able to construct a replica of existing screens automatically from CASE information, because the full functionality of the service(s) is available immediately. The most urgent improvements can then be worked on, leaving the rest unchanged.

As an example, the standard Dialogue Manager product can generate automatically Visual Basic screens and processing code from VME TPMS dictionary information, thus giving a Windows look to existing TPMS screens. In addition:

- all the generated controls are linked to their appropriate definitions in the inventory file. This would allow, for example, fields to be moved around on a screen, without the need for code to be written.
- the processing code automatically generated is in source form, so that the degree to which processing is automated can be controlled field by field, and/or screen by screen.

3.5 Scaleability issues and export inventories

One of the main advantages of mechanical extraction and generation tools is that they allow services with hundreds or even thousands of screens to be tackled. However, to achieve this in practice, it may be necessary to split the applications solution into a number of separate cooperating applications. The main reasons for this are:

- the need to allow many developers to work on the same applications
- the limitations of various HCI toolkits in the number of screens they support.

Dialogue Manager supports relationships across applications by the use of export inventories, as illustrated in Figure 4.

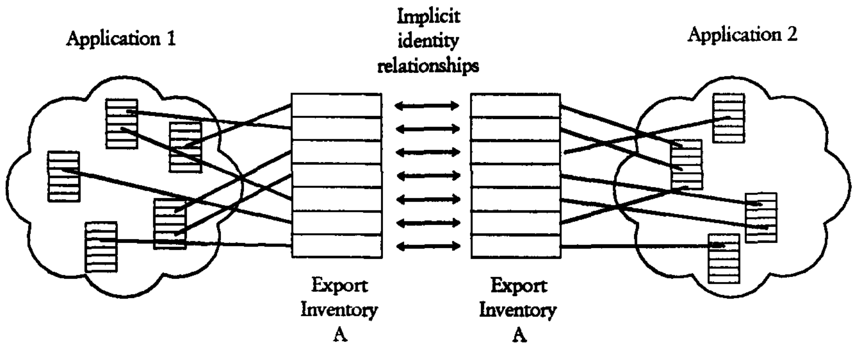


Figure 4 Export Inventories

Each application has its own data definition file. Data sets (inventories) are private to that application. However, export inventories define data sets which transcend applications. Dialogue Manager links export inventories with the same name in different applications as if each item had an identity relationship with all its equivalents. The linkage may be many-way if required.

The effect of this linkage is that if an item is set in an export inventory in one application, the same value appears for the same item in all the other applications. If an event trigger is defined for that item in one application, which is to be fired when that item is written to, then it will fire when the item is written to in another application.

Support for multiple cooperating applications also has implications for service access. For example, one application sends a particular screen to a service. However, the screen received in response is not necessarily in the set supported by that application. While the runtime structures generated for service access in different applications are separate, the Dialogue Manager core can "see" all of them, and can therefore identify which application should process a particular screen. Interfaces are defined so that the original application is informed that another application needs to process the received screen, and so that the identified application can be told to process it.

4 Dialogue Manager in action

Dialogue Manager was first released generally in December 1992, with a second release in September 1993. A third release is planned for mid 1994. It has been adopted by a number of large organisations, both to construct new composite applications and to integrate a number of services via a single modern user interface.

Some of the productivity gains have been very dramatic. In one case, the operator time needed to assemble relevant customer information to respond to a telephone enquiry was reduced by a factor of twenty five.

The development process has also proved very productive and there are now enough examples to allow development effort to be predicted with some confidence. It has often been possible to produce an initial subset development to achieve real business benefits both quickly and cheaply. As well as providing initial estimates of cost, the resultant improvements in productivity sell themselves within the organisation in a way that proposals cannot [Peters, 1988].

The scaleability issues addressed in section 3.5 have proved important - in one case, a composite application has been constructed from nearly 30 separate applications.

Data mapping has allowed the HCI to be designed as the business requires rather than be constrained by the way in which the services provide data. The screens displayed are not merely representations of existing screens. Indeed, it may require some thought to relate them visually at all.

Last, but not least, no change has been required to any existing service, and in each such case, a design based on Dialogue Manager has been deployed without disruption to existing users.

Acknowledgements

The authors wish to express their thanks to all those who have been involved with Dialogue Manager, whether in development, marketing or management.

Trademarks

TUXEDO is a registered trademark of UNIX Systems Laboratories, Inc., in the U.S.A. and other countries.

MICROSOFT is a registered trademark of Microsoft Corporation in the U.S.A. and other countries.

VME and Vision-master are trademarks of International Computers Ltd.

References

BRENNER, J.B. Client-server architecture, *ICL Tech. J.* Vol. 9 Iss. 1 pp. 3-17, 1994.

GARTNER: Software Management Strategies E-400-1121, 1992.

ICL: Dialogue Manager Reference Manual ICL Reference number 52381/003, September 1993.

PETERS T.: Thriving on Chaos, Macmillan 1988, Part III I-3.

PROKESCH. S.E.: Mastering Chaos at the High-Tech frontier, an interview with Silicon Graphics' Ed McCracken, Harvard Business Review Nov-Dec 1993.

Biographies

Roger Thompson

Roger Thompson joined ICT in 1967 prior to reading Mathematics at Cambridge University. He worked on systems software development for ICT 1900 systems, then as chief designer for operating system development on ICL 2903 and ME29.

More recently, he has represented ICL on working groups of X/Open and UNIX International concerned with PCs and User Interfaces.

In 1991, he co-founded Business Opportunities Development with Ian Robertson, which was set up to identify business opportunities and develop innovative products quickly to exploit them.

Ian Robertson

Ian Robertson has over 25 years experience of systems design and project management with ICL and its predecessors.

He was involved with UNIX systems from their introduction into ICL in the early 1980s and represented ICL on X/Open Networking and Transaction Processing technical committees.

His area of technical specialisation is Transaction Processing (TP) with which he has been involved for over 20 years. Immediately prior to co-founding Business Opportunities Development he was responsible for TP strategy for ICL mid-range systems.

Distributed Printing in a Heterogeneous World

Steve Hilditch
Client-Server Systems, ICL, Bracknell, UK

Abstract
Today's computer systems are becoming more complex, more heterogeneous and more structured towards the client-server paradigm. Print spooling is also faced with the challenge of *printing at the point of need* and the opportunity of newer, low cost printers. All these changes are happening within the context of a continual need to reduce systems administration costs. Therefore, there is an increasing need for an easy-to-use, fully heterogeneous, client-server, distributed print spooler. This paper discusses the requirements on such a spooler from the point of view of an IT manager and presents a distributed architecture. Current ICL printing products are also discussed.

1 Introduction

Three important trends are making distributed printing more and more necessary in mid to large computer systems:

- the movement to networks from centralised processing
- the desire to print at point of need
- the availability of low cost, low throughput printers.

It is becoming common to have a variety of computers within an enterprise computer system. There can be a variety of manufacturers, a variety of platforms and a variety of operating systems, each with its own spooler. Similarly, the growing desire for a variety of printers in a variety of locations, coupled with increased low cost printer availability makes the situation more complex. The need for well-defined, well-integrated distributed printer systems management software is growing.

1.1 What is Distributed Printing?

Distributed printing can be defined as the ability to submit, route and control print jobs within a network of possibly heterogeneous computer systems and possibly heterogeneous printers. On an individual level, this means being able to submit a print job from anywhere to anywhere, maintaining control of it after submission, and being kept aware of printers being taken off-line and paper outs. On a corporate level, this means having the capability of controlling all the connected printers, collecting usage statistics for planning and billing, and controlling printer servicing centrally.

1.2 Output and The Paperless Office

Since offices are exchanging more and more electronic information, a paperless office is becoming more of a reality. This raises another important aspect of printing: its relation to output in general. It is outside the scope of this paper to discuss output at length but it will be touched upon at various stages.

Output is the sum of all data exported from a computer system. This can take many forms:

- printed hard copy
- EDI (electronic data interchange)
- BACS (the banks' automated clearance system)
- IPM (inter-personal messaging or electronic mail)
- FAX.

The forms of output listed above are all essentially **one-way**. Output is not interactive communication such as remote login or video conferencing. Output is essentially taking an object of revisable format and transmitting a non-revisable copy of it.

The differences between the types of output are reflected in their ***usage*** and ***format***. Printed hard copy can be used for informal or legal purposes of which faxes are the express variety, IPM for informal communication, BACS for account transfers, and EDI for business transactions. Each of the types of output has its own text format, its communications protocols and media. It is possible to use common electronic communications such as X.400 as a basis for more than one form of output. It should be the long term aim to integrate as many forms of output as possible in order to increase ease of use for the end user and the systems administrator. An important part of this integration would be the ability to group more than one type of output together, e.g. an invoice with an informal covering letter. As regards printing, the distinctive features in relation to other forms of output are that its medium is paper, that its format is a human readable language, and that printing may be for use within the enterprise as well as outside the enterprise.

1.3 Other Literature

A suggested implementation of a distributed print spooler, **nlp**, is described in [Fletcher, 1992] based on TCP/IP and the Berkeley UNIX™ LPD protocol. **lpd** was developed on-site in the SAS Institute Inc. where the computers in the network system vary widely. The industry standards for distributed printing are ISO 10175 DPA (distributed printing application) [ISO, 1993] and POSIX P1003.7.1 [POSIX, 1993]. These standards are discussed and elaborated in the proposal of [Duvall, 1994]. This paper aims to be more implementational than the standards documents mentioned above, but at the same time more embracing than the paper of Fletcher.

1.4 Overview

The complexity of the distributed printing subject and the limited length of this paper dictates that the paper's scope be limited. Therefore, it concentrates on those issues of most relevance to IT managers. This paper considers first the requirements on a distributed print spooler from the point of view of IT managers, and then describes a distributed print management architecture. The following section describes some current spooler products and a spooler which most clearly reflects the proposed architecture.

2 Requirements

The requirements outlined below are designed to be those of most relevance to IT managers.

2.1 Interoperability

The first key requirement is that the spooling facilities on one system be able to cooperate with those of another system. This means being able to exchange print jobs and to control information. There are three main aspects to interoperability:

- different vendors
- different platform architectures
- different operating systems

The situation is made worse by the provision of separate print spoolers within certain applications. Those applications were perhaps written in the days when the native print spoolers provided with the operating system were not as user friendly as they might have been and they perhaps lacked facilities now considered important. However, the situation in many enterprises is a widely distributed heterogeneous system. Developing systems management tools, such as a distributed spooler, in such an environment is a challenge. Standards are an important aid to the development of solutions and they provide the long term solution, but the key to today's distributed spooling is in the hands of the ISV (Independent Software Vendor) or solutions provider: integrating what is now available.

2.2 Devolved Responsibility and Printing at the Point of Need

The trend towards devolved responsibility within an enterprise means that the end user increasingly wants to be able to print at the point of need. This means firstly that print jobs should be able to be sent to any printer in the network, assuming the correct privileges and the appropriateness of the printer. It is desirable that certain forms of print jobs appear as close to the job submittor as possible for personal collection, e.g. printing slides for a presentation. Printers can no longer all be gathered in central places; printers will have to be connected to LANs, departmental servers and PCs. Secondly, there is a need for employees to be able to login from another site, perhaps when visiting, and be able to print close to where they are. These two requirements can be summed up in the maxim "submit from anywhere, print to anywhere". The flexibility of print management software implied by employee empowerment in turn dictates that all the print spoolers, from mainframes to PC-LANs, be integrated into one print management system.

2.3 Centralised Administration versus Devolution

In a world of increasing system complexity, the ability to control a computer system either centrally or de-centrally is an important consideration. An administrator should be able to gather usage, performance and failure information easily, on the enterprise level or departmental level, despite the complexity and location of the network components. In this way, billing, capacity planning and the management of system servicing can be performed efficiently. Centralised mainframe systems are easily controlled centrally and PC-LANs are easily controlled by individual users. However, with the introduction of complex heterogeneous computer networks, the ability to support systems administration at the appropriate level of devolution has become much harder.

In relation to printing, certain administrative tasks can naturally be decentralised, for example, re-filling paper on printers. On the other hand, other administrative tasks are better centralised, for example, security, billing and capacity planning. Print management software requires the gathering of management information about the various printers and print queues across the network. It also implies the ability to install, configure, start, stop and maintain distributed spooling. Therefore, print administration can only be achieved by close integration of all the spooling facilities.

2.4 Open Standards

Openness and standards are the long term solution to interoperability and heterogeneity. As the interfaces are refined and the solutions providers use those interfaces, so integration of multi-vendor networks becomes relatively straightforward. The pioneering work was done by the European Computer Manufacturers Association (ECMA) upon which the International Standards Organisation (ISO) standard ISO/IEC/DPA 10175 [ISO, 1993] is based. ISO 10175 consists of two parts: the Abstract Service Definition Parameters, and the Protocol Specification. The former section describes the distributed printing application (DPA), its object classes in object oriented terms, the processing model, the operations and the object attributes. The latter section describes how the distributed print service modules in different locations can communicate. ISO 10175 is at International Standard status. The Portable Operating System

Interface (POSIX) standard POSIX P1387.4 [POSIX, 1993], formerly P1003.7.1, is very much based upon the ISO 10175 standard but moves towards an implementation by incorporating most of the features of the Palladium print management system from MIT. The POSIX standard should be able to be endorsed in 1994. Work is under way to draft an X/OPEN standard from these other standards. It is the responsibility of print management implementors to provide input to these standards and then follow them as soon as possible after they have been endorsed.

2.5 Cost Reduction

In a world of continuing reduction of unit costs, distributed print management needs to be designed to reduce the following costs:

- running costs
- development costs
- upgrade costs
- operator intervention
- training costs

Printing must have interfaces to the end user and the administrator that are easy to use. Tools will have to be provided to aid print management software development, capacity planning, installation and configuration. Administration must be as mechanised as possible, to reduce the number of mundane tasks of a systems administrator and simplify training.

2.6 Client-Server Design

In a distributed world, it is becoming increasingly important to decentralise the processing of all tasks. Print management is no exception. Application distribution by the use of client-server architectural methods has three major benefits:

1. *performance*, by best utilising the various hosts.
2. *flexibility*, by easing system reconfiguration and system upgrades.
3. *resilience*, by duplicating services to cope with individual failures.

2.7 Environmental Friendliness

Environmental issues are becoming increasingly important, not just to save money by re-cycling but also to satisfy our consciences in this resource-draining civilisation. It is often the rank and file employees who drive the corporate green conscience and therefore prowess at recycling can increase enterprise morale. Some of the most pressing environmental issues associated with printing are:

- duplex printing
- recycled paper
- toner refilling

- printer component recycling
- the growth of the electronic office
- ozone reduction.

3 Architecture

3.1 Hardware Topologies

Figure 1 shows the numerous variations within a networked print environment. Distributed print clients can be attached to the network directly as a PC or they can be connected indirectly via a departmental server, a terminal concentrator or a mainframe. The printer they wish to use may be directly attached to the back of their PC or another PC across the LAN, or may be attached to a departmental server, or a dedicated print server, or a terminal concentrator, or even directly to the LAN. LAN-connected printers usually contain communications software for a variety of network protocols, but can also contain part of the print management software or act as Simple Network Management Protocol (SNMP) agents for operations management software.

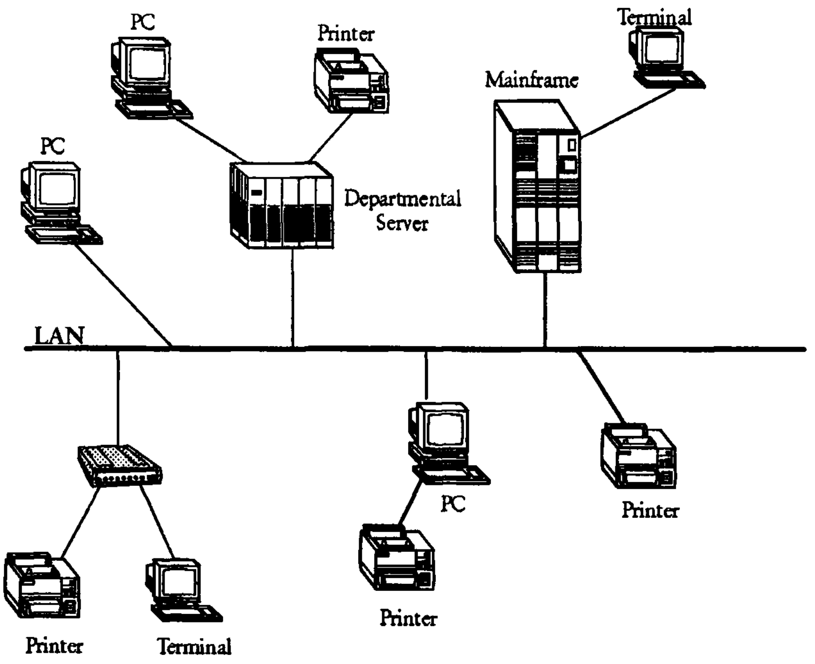


Figure 1 Hardware Network Topologies

It is this final category of LAN-connected printers that are being promoted by printer manufacturers such as HP and QMS. They point out that LAN-connected printers have a number of advantages that justify the major disadvantage of a higher price:

- *Speed*: LAN bandwidth is greater than serial or parallel cables. However, the printer may itself be the performance bottleneck.
- *Flexibility*: the printer is able to handle a number of different LAN protocols simultaneously: OSI, TCP/IP, IPX, AppleTalk, NetBios, etc.
- *Resilience/Availability*: a LAN-connected printer is no longer dependent on the availability of the server to which it is attached.

3.2 Two Classical Approaches

There have been two classical approaches to print spooling:

- the centralised mainframe approach
- the decentralised PC-LAN approach.

The mainframe print spoolers centred all the printing facilities around the host. The user on a remote terminal submits a print job, the job is kept on the host until being sent to a printer local to the host. Since all the jobs are processed by a single central spooler, systems administration is kept simple. The central location of the printers used to cause inconvenience until the introduction of terminal connected printers, such as Direct Print on VME.

On the other hand, the PC-LAN spooling approach is to decentralise the whole process. The end user submits a job, perhaps with data from the PC, the job is queued either by the same PC or by a departmental server and the hard copy appears either on a printer attached to the original PC or on a printer in the department. End users can export their own printers as a service to others. This implies that the end user administers the spooling process and the printer servicing.

PC-LAN spooling means:

- Increased ease of access for the end user.
- Increased administration skill required for end user.
- Devolved administration and responsibility.
- Difficulty in centrally collecting printer statistics, fault messages.
- Difficulty in centrally billing printer usage.

3.3 Spooling as "Middleware"

The first step towards a distributed printing architecture is to see printing as middleware. If, for example, printer usage and billing are to be administered centrally, all jobs will have to be submitted through a common spooler. The spooler sits above the operating system and networking which, in turn, sits above the physical system, see figure 2.

The spooler provides interfaces to end users, applications and administrators. The figure also shows the interface provided by the application to the end user. End users, administrators or applications could call the native operating system print spooler if desired but this activity would not be managed by the common spooler.

Spooling as "middleware" implies that both users and applications submit their jobs through a common spooler, instead of calling the printer drivers directly. Spooling as middleware also implies that both an end user interface and an application programming interface (API) be provided by the spooler. This does not imply that all jobs must be submitted to a central point, only that common and interworking spooler components be used throughout the network.

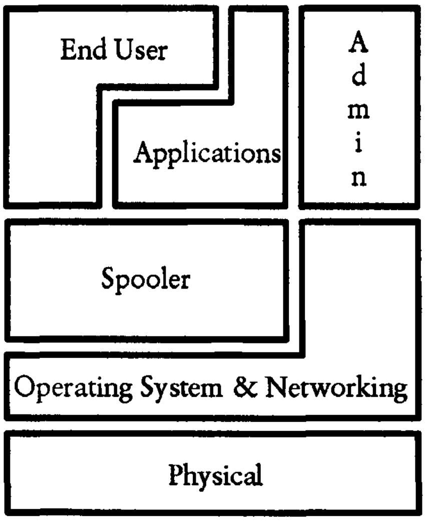


Figure 2 Spooling as "Middleware"

3.4 Print Job Dataflow

A distributed printing architecture must be able to transmit print job data from anywhere to anywhere. This means that the print job data may have to be formatted, filtered, transmitted, translated and stored in a number of places before reaching output. See figure 3.

The first stage of processing the print job data is to create the user-desired print job data. The input data is in a revisable format and an un-revisable copy is taken of the required file or part of file. This is most naturally done by application software on the client computer system. At this stage a front sheet or separator may be added.

After the print job data has been created it is possible that some other software may post-process the print job data, splitting the data into multiple print jobs with different subsets of the print job data. The splitting is indicated on the figure as a double-headed sideways arrow. This form of print job data splitting is common in large batch print jobs.

The second stage consists of formatting or translating the print job data before transmission across the network. This might involve translating into PostScript or PCL (HP Printer Control Language).

The third stage is to store the data. For reasons listed below, it is best to store the print data close to its original source.

The fourth stage of processing is after a printer has been chosen for the job, either explicitly by the end user or automatically by the spooler. A transmission route to the printer is selected and the data is moved across the network. Note that the print job data format may have to be changed en route. For three reasons, transmission is best delayed, called **late binding**, until the printer is ready to accept the job:

1. It avoids possible unnecessary copying of the data if the destination printer is local to the data source.
2. It allows the decision about the printer to be delayed if load balancing a number of printers.
3. It allows easy recovery after printer blockage, either failure or paper out.

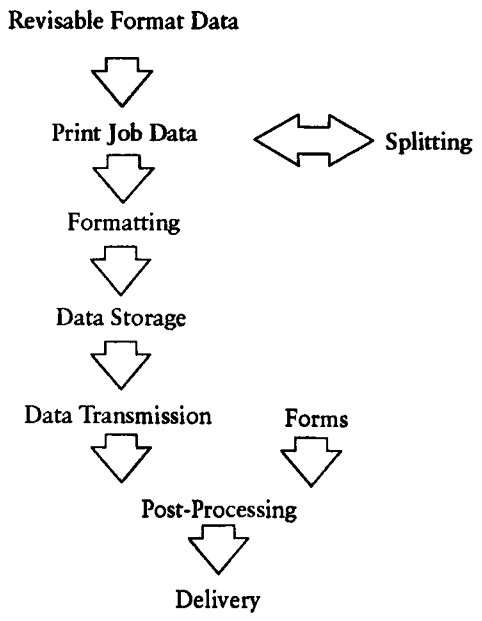


Figure 3 Print Job Dataflow

The fifth stage is post-processing, which is done close to the printer. Downloaded fonts and output forms can be added, standard printer control languages (e.g. PCL or PostScript) can be emulated, and the final output assembled.

An optional final stage is delivery of the output from the printer to its required destination. This may be achieved by an operator using the front sheet or separator. In a large enterprise with specialised centralised printers, automation of this stage is crucial.

3.5 Client-Server Structure

The internal structure of a distributed printing architecture determines its effectiveness:

- efficiency
- flexibility
- manageability
- ease of use
- resilience

A distributed printing architecture has five essential logical components:

- user interface
- data storage
- scheduling
- printer control
- recovery from server failure.

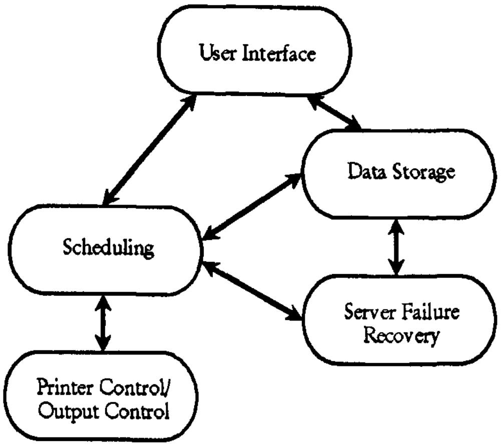


Figure 4 Client-Server Structure

Each component provides a service to end user clients or other components. See figure 4. The arrows shown indicate communication between the various components. In each of the internal communications, one component is a client and the other is a server. Each of the components can be implemented by a number of instances across the network in order to provide flexibility, speed and resilience.

3.5.1 User Interface: Client

User access (either end user or administrator) is through either GUI, commandline, C library or character-based interfaces. The user interface checks printing privileges, adds any default parameters and executes the pre-processing of the print job data. The user interface code executes most reasonably on the host nearest to the job submittor. Its appearance may vary according to the terminal and operating system used (e.g. character-based, X windows or Microsoft® Windows). However, there should be no necessary restrictions on a particular client's functionality.

For the end user, some or all of the following are appropriate:

- browsing suitable print queues to which access has been given
- submitting jobs
- checking the status of jobs
- suspending the job whilst in a queue
- removing a job from the queue

For the administrator, some or all of the following are appropriate:

- monitoring printer status and queue status
- adding and deleting privileges, printers and queues
- managing recovery
- managing load balancing and capacity planning
- managing servicing and billing

The user interface/client component will interface with the Data Storage component to store the print job data, and with the scheduling component managing the queue selected for the job.

3.5.2 Data Storage

The data storage component manages the storage of the print job data, the print queues, access control information and the print configuration information. Its function is different from that of the scheduler's since it is concerned only with safe storage of information and not with job dynamics. The data storage component provides a service to the scheduling component, that is storage. Resilience to failure can be achieved by duplicating the print job data on different storage media accessible by more than one host. After failure, all information about configuration, permissions, current print jobs and current print job data would be immediately available in order to carry on spooling.

3.5.3 Scheduling

The scheduling component controls the print queues, determining priority, destination and transmission path. Each scheduling component instance may manage a number of print queues, and thereby a number of printers. The queues can be kept safe by duplicating storage on more than one medium, accessible from more than one host. The scheduling component interacts with the printer control components associated with its print queues. Load balancing can be achieved by late binding: delaying the sending of a job to a printer until it is about to be idle. Late binding also aids recovery from printer failure, printer off-lining and disconnection from a printer, by re-routing the job to another printer.

3.5.4 Printer Control

There is one printer control component to manage each printer. It will initialise and configure the printer, initiate the post processing of the print job data, call the relevant printer drivers and manage the error messages from the printer. A printer control component responds to job requests from the scheduler and returns messages indicating job-done, ready-for-next-job and error messages. The printer control component associated with a printer most naturally executes on a host nearest to that printer.

The printer control component could be replaced by a more general *output control* component in order to bring other forms of output within the distributed architecture. The output control component would be responsible for communicating the output from the computer system: either managing the sending of an electronic message or dialling a FAX machine and sending the data.

3.5.5 Server Failure Recovery

The server failure recovery component monitors the progress of the other print management software processes and their hosts. It is responsible for the initialisation of the processes and their restart after failure, either automatically or manually. The important data structures of failed processes can be made accessible even after medium or host failure, thanks to the Data Storage component. These surviving data structures can be used by the replacement process. The server failure recovery component can be implemented as a number of instances, all monitoring each other. In this way print spooling can be made resilient to network failure, printer failure, process failure, medium failure and host failure.

4 Implementations

The spoolers available in the marketplace usually fall into one of two categories:

- the centralised spooler
- the decentralised spooler.

4.1 Mainframe VME Spoolers

In the centralised spooler category fall the mainframe spoolers, e.g. on ICL hosts, the VME print spooler and the Gandlake spooler. The printers are normally LAN-connected, control is centralised and client-server splitting of print management is minimal.

4.2 Microsoft LAN-Manager

The Microsoft products: LAN Manager, Windows For Workgroups and Windows NT™ allow PC owners (386 or later) to export the services of a printer attached to their PC by creating and exporting a print queue. The management of the print queue remains their responsibility, although it is not an onerous task, and they have the power to suspend and restart the print queue. Anyone using the LAN can *connect* to the print queue providing they have the permission (an optional password associated with the print queue). Having connected to a print queue, its contents are visible. In this way an end user can browse those connected print queues in order to choose a lightly loaded printer. Windows NT does allow print queues to be administered centrally and also allows print queues from more than one LAN Manager domain to be administered centrally. There is as yet no spooler component that enables the spooler to recover from host failures.

4.3 Novell NetWare

Novell NetWare's print spooler is split into two major components and three print utilities. Print jobs are written to queues (subdirectories on file servers), these are polled by a small number of printer servers: PSERVER, which are perhaps executing on a host different from the file server. The PSERVER instances send the next highest priority job to the printer (or the next available printer if a set of printers is specified). Each printer is managed by RPRINTER software (called NPRINTER in version 4), perhaps running on the PC to which the printer is attached, perhaps running on the printer itself, as in some LAN-connected printers. The utilities provided: CAPTURE, ENDCAP and NPRINT enable non-NetWare compatible applications to submit print jobs. CAPTURE and ENDCAP redirect output from local PC printer ports to a print queue, whereas NPRINT redirects files for printing. According to the proposed architecture, RPRINTER (or NPRINTER) does printer control, and PSERVER does scheduling, but there is no resilience to host failure when that host is running the relevant PSERVER software.

4.4 Standard UNIX Spoolers

The standard UNIX spoolers: SVR4's **lp** and BSD's **lpr**, provide certain distributed spooling facilities. A UNIX host can export access to its print queues to other UNIX hosts. This allows jobs to be sent across the network. However, the interface is commandline and the location of the printers is not made transparent to the end user. These ease of use shortfalls for end users are also reflected in the lack of easy to use administration tools.

4.5 Siemens-Nixdorf's Xprint: From UNIX Outwards

ICL's PrintManager on DRS/NX, UnixWare and SCO UNIX, which is a port of the Xprint spooler from Siemens-Nixdorf, can be seen to be a good starting point for a distributed print spooler implementation. Judged by its architecture, it provides the desired features set out in the architecture section above:

- **interfaces:** as well as graphical end user and administrator interfaces, and a commandline interface, it provides both **lp** (UNIX SVR4 spooler) and **lpr** (BSD UNIX spooler) interfaces. In addition, there is an API for developers.
- **client-server:** Xprint is composed of five modules that correspond to the five essential components of a distributed print spooler architecture. It is unique in providing a server failure recovery module to monitor the other print management processes and recovery from loss of hosts as well as printers.

Although restricted at present to UNIX variants and Siemens-Nixdorf mainframes, it is being ported to UnixWare 2.0, NetWare and Windows NT. In addition, client packs are available for DOS and Windows PCs.

In order to provide a distributed enterprise print spooler, Xprint will need to be integrated with native mainframe spoolers as well as existing legacy applications, such as OfficePower.

5 Conclusions

In this paper we have reviewed print management in terms of IT management requirements and in the wider context of output. A distributed print management architecture has been proposed:

- considering print spooling as middleware
- involving six stages of the print job dataflow
- consisting of five crucial software components

It has been shown that the current ICL product PrintManager (Xprint from Siemens Nixdorf) satisfies that architecture completely, unlike other print spoolers considered.

At present it is possible to network printers in such a way that an end user can submit a job from anywhere to any printer, but not all end user features would be available (e.g. stopping a job after queuing may be impossible). What is clear is that administering a distributed spooler in a heterogeneous network is currently a difficult task, although Xprint provides a solution in the UNIX environment and in an environment that is becoming ever more heterogeneous. A complete distributed print management product is not yet available in today's heterogeneous world, but the subset of that world that can be integrated is growing. Standards will define the end implementation when they are agreed and then followed.

Acknowledgements

Thanks are due to Bernt Palmqvist, Barrie Archer, Ann Knowles, Dave Hollingsworth, Wojciech Kierstan and Alex Sharp of ICL for helpful information, also to John Pinkerton and Barrie Archer (again) for giving detailed comments on the draft version of this document.

References

[Duvall, 1994] Distributed Print Resource Management & Interoperability Standardization, Keith E.Duvall, IBM, submission to X/Open's Systems Management Working Group, 1994.

[Fletcher, 1992] nlp: A Network Printing Tool, Mark Fletcher, SAS Institute Inc., USENIX LISA VI, October 19-23, Long Beach, CA, 1992.

[ISO, 1993] ISO/IEC/DPA 10175, International Standards Organisation, 1993.

[POSIX, 1993] System Administration Interface/Printing P1387.4, Institute of Electrical and Electronics Engineers, Draft Standard for Information Technology - Portable Operating System Interface (POSIX), 1993.

Trademarks

UNIX is a registered trademarks of UNIX Systems Laboratories, Inc., in the USA and other countries.

MICROSOFT is a registered trademark and Windows NT is a trademark of Microsoft Corporation in the U.S.A. and other countries.

Biography

Steve Hilditch

Steve Hilditch gained a PhD in pure Mathematics (Algebraic Topology) and a BA in Theology before turning to Computers in 1987. From 1987 until 1991 he worked as a research associate and part-time lecturer at Manchester University on joint projects with ICL, West Gorton: Flagship, EDS and *GOLDRUSH*. During this time he completed his MSc in Computer Science. After a year learning French, Steve joined ICL, Bracknell. He now works in Server Systems Division within ICL, Client-Server Systems. Since joining ICL full-time, Steve has filed 8 patents in both hardware and software.

Systems Management: an example of a successful Client-server Architecture

Mike Small and Dave Roberts
Client-server Systems, ICL, Bracknell, UK

Abstract

The need for remote management of distributed systems has led to the development of a client-server architecture for this purpose. This paper describes ICL's Community Alert Management which is used to implement such an architecture, together with some practical examples of its use.

1 Introduction

The aim of systems management is to enable service providers to manage distributed, complex, multi-vendor information systems.

Systems management has a broad scope which covers all of the tasks necessary to ensure the availability, performance, security and change of IT systems within an organisation. Traditionally, tools and facilities for systems management have been provided locally as part of proprietary architectures for the management of particular manufacturers' equipment. By contrast, network management developed in response to the need to provide remote control over equipment which is geographically widely distributed and which is from a number of suppliers.

ICL's systems management architecture evolved [Gale, 1991] from the work done in the early 1980s to build and manage large scale multi-vendor systems for ICL's large customers in a number of areas including retail, government and public services. This led to the definition of ICL's OPEN*framework*™ architecture and the development of a number of products. These products include the Open Systems Management Centre (OSMC) and Retail Systems Management.

Over a period of time, various components of this architecture have been implemented. One key component is the management messaging system Community Alert Management (CAM). The software which implements this runs over the X/Open* Transport Interface (XTI) and provides many interesting features which are described in this paper.

Finally some of the live applications of CAM are covered.

2 Architecture

2.1 General Management Model

ICL's systems management architecture is based on ISO and de-facto standards. The basic model for management, adopted by OSI, comprises a managing system and a managed system as shown in figure 1. The managing system contains a managing process which exchanges management operations and notifications with an agent process within the managed system. The managed system contains one or more managed objects and the agent process is responsible for mediating between what is communicated in the management protocol and what happens in, or to, the managed object.

The attractiveness of this open management model is that it provides an effective and efficient means of managing a wide variety of remote platforms in a standards conformant way.

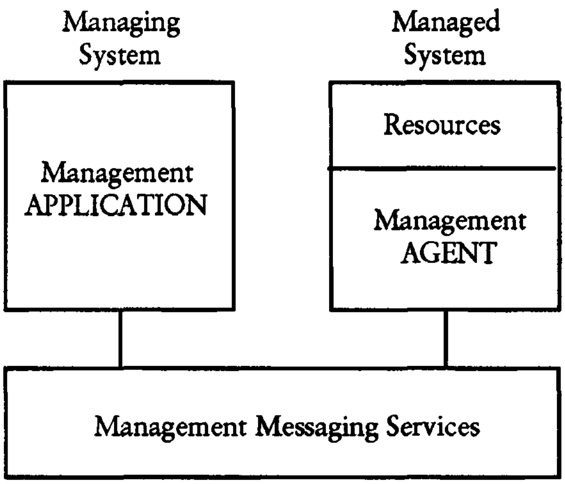


Figure 1 OSI Management Model

2.2 Client-Server Approach

The nature of the model described above is such that organising it as a network of clients and servers is very natural, and also a convenient approach as we try to show in the rest of this paper. In this approach both the managed systems and the managers are clients of a set of management services. These services provide a flexible but standard way of communicating with, and operating on, different types of system.

As pointed out by [Brenner, 1994] a client-server system is not necessarily restricted to one client interacting with a single server.

There can well be several servers providing different services at the same or different times to one or more clients. The clients and the servers can also be distributed throughout the network as is illustrated in figure 2. Here both the manager and agent are clients of a messaging service. To send a message the agent passes this to the local message server, which is in turn a client of other network services such as TCP/IP. The message server then passes the message to the manager over a route, hidden from the agent and the manager, which may involve other message servers and a variety of network services.

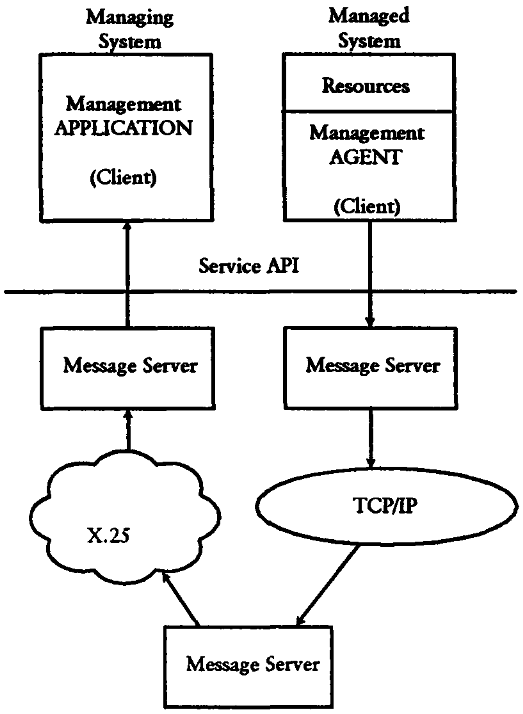


Figure 2 Messaging Service

One benefit of a client-server approach is that the managed systems need not be sized to provide all of the management functionality themselves. This functionality may be located wherever it is most effective or efficient. For example, event routing, filtering and correlation services may be provided at the centre so that managed systems need only concern themselves with reporting their own events.

The client-server approach also makes it possible to distribute certain services. For example, file transfer services may be located at key points in the network so that only small numbers of copies of software need be distributed over the wide area network. This makes for a scalable solution which is capable of managing very large numbers of systems.

2.3 Management Infrastructure

The remote management of systems requires connection services between the managed systems and the management centre. These services form a management infrastructure providing:

- management messaging
- bulk data transfer
- interactive access

2.3.1 Management Messaging

This covers relatively small amounts of data involved in communicating events, alerts, alarms and commands.

The established ICL protocol for management messaging is Community Alert Management (CAM). This has been in use since 1986 and runs over both OSI and TCP/IP transport infrastructures. CAM provides a number of useful features which are described in more detail later in this paper:

- *Real Time*: CAM works in real time so that messages are delivered swiftly enough for operational monitoring and control.
- *Routing*: CAM uses the name of the target object to determine the route to send a message.
- *Resilience*: CAM is able to re-route messages avoiding failed paths.
- *Filtering*: CAM provides facilities to send messages selectively to certain destinations depending upon the type and content of the message.
- *Recovery*: CAM can retain messages for destinations while no path is available to send and to re-send these messages when the path becomes available.
- *One to Many*: CAM can broadcast a single message to a number of destinations.
- *Store and Forward*: CAM is able to deliver messages when the final destination is not known directly to the originating system.

CAM is available on all ICL Open platforms as well as a range of non ICL platforms.

2.3.2 Bulk Data Transfer

This covers the large amounts of information communicated periodically as part of software distribution and configuration changes.

2.3.3 Interactive Access

This provides terminal access connections required by remote login.

3 Implementation

3.1 Community Alert System (CAS)

The CAS software implements the CAM messaging system mentioned in section 2. CAS enables elements of a distributed system to send and receive management messages in real time. CAS is based on the use of the X/Open Transport Interface (XTI).

Each managed system may have one or more CAS service and each CAS service is identified by an address known as the CAS address. Each CAS service may provide messaging facilities for a number of applications each of which are identified by a CAS sub-address. By means of CAS each of the applications may send and receive messages.

3.2 CAS Components

The overall structure of CAS is shown in figure 3.

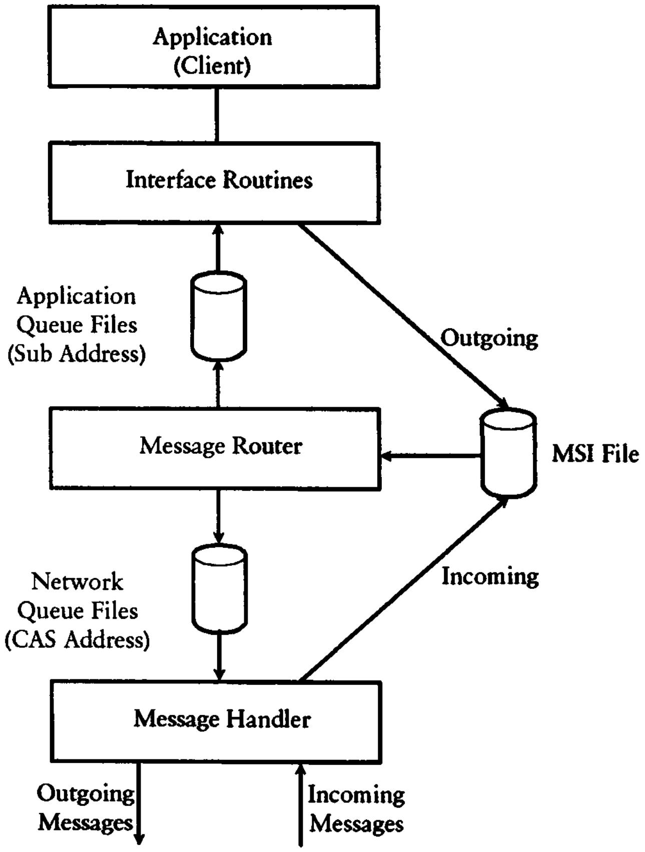


Figure 3 CAS Components

3.2.1 Message Router

This is a software module which takes messages from a queue and decides the action for each one. Each message is dealt with on a first come first served basis.

A configuration file, known as the routing file, determines how messages are to be dealt with. Rules may be set up whereby actions may depend upon the message type, message content and addresses. Actions taken may include:

- transmission to another network element
- passing the message to a local application
- logging the message
- discarding the message, with or without a non-delivery indication to the originator.

Rules are used to divide the messages into logical groups which may then be selected for particular actions. The rules are written as boolean expressions using the names of various fields making up messages, constants and boolean operators. The fields include:

MT	Message Type
DATE	Date
TIME	Time
OAT	Overall Alert Type
MAC	Major Alert Cause
SC	Specific Component
AC	Alert Code
CAS	CAS Address
SUB	CAS Sub Address
DET	Details
RET	Retry Field

The operators available are:

=	Equals
NE	Not Equals
<	Less than
>	Greater than
>=	Greater than or equal to
<=	Less than or equal to
AND	Logical and
OR	Logical or

The *Message Type* identifies the style of messaging being employed. The most common style is the 'Directed Message', where the originator identifies the recipient. Routing of such messages is to send to the recipient, unless the routing rules decide otherwise.

Alert messages are also catered for. Here, an application posts a 'cry for help' to CAM and the *Message Router* decides who is competent to assist, based on the characteristics of the message. Alerts do not identify their recipient. This conveniently separates the concern of raising an alert from that of deciding what to do about it.

Since it is the responsibility of the message router to route the alert appropriately, the application relies on CAM to deliver the message suitably. For instance, should the Router decide that the message is of no significance to anyone else and discard it, then the originator need not be concerned.

However, directed messages are treated very much like mail. The default is to deliver at all costs, however long it takes. To continue the mail analogy, it is equally possible to mark the envelope with 'if undelivered return to sender' or 'if undelivered do not return to sender'. Again, these are defaults, set by the submitting application and the message router is at liberty to override these rules if it thinks fit.

A message may not be delivered if its destination cannot be reached immediately. For local destinations, the router knows which applications are connected to CAM. For remote applications, discard is done in cooperation with the Message Handler which evaluates whether communication failures are transient or more long lasting.

3.2.2 Message Handler

This module communicates between the network and the message router described above. It ensures that each message is successfully transmitted to its destination using the path specified. This software uses the X/Open Transport Interface to the network transport services.

The *Message Handler* maintains a queue for each remote destination. Message receipt is confirmed by the receiving message handler before a message is discarded from the queue.

Note that whilst CAS provides application-to-application messaging, it obviates the need for direct application-to-application communication links. All intra-system communications are between CAS message handlers.

Message Handler passes all incoming messages to message router, which may deliver them to local applications and/or onward route them via the message handler.

3.2.3 Message Storage
In principle, messages are held in files thus avoiding their loss when processing or transmission is held up for any reason. Messages entering the system, whether from local applications or from the network, are held in the *Message Submission Interface* (MSI) directory until the message router is able to deal with them. Messages output from the message router are placed in files to await transmission across the network by the message handler or processing by local applications. Messages in these files are processed on a first in first out (FIFO) basis.

For efficiency, message passing may be via direct connections between the message router and local applications or the message handler. In such cases, messages are secured in files only if a connection becomes congested or if necessary to avoid loss.

3.3 Availability of CAS
The CAS software described above is now available on a range of mainframes and UNIX® systems. The mainframes include ICL VME® and IBM MVS (running under CICS). The UNIX systems include ICL DRS 6000 and DRS 3000 series under DRS/NX, Hewlett Packard HP9000 series under HP/UX, SUN systems under SUNOS and Solaris 2, SCO UNIX, Pyramid Systems and DG Aviion. CAS has also been implemented on other regimes including OS/2 and VMS.

4 Examples of CAS in use

The CAS software, which forms part of a number of different products, is also used by certain customers directly.

4.1 OSMC Operations Manager
OSMC Operations Manager caters for the remote management of a network of systems [Hacker, 1991]. It provides a comprehensive view of the overall system and the services provided. A high quality graphics display [Small, 1991] shows a map of services and servers in the network and their real time status. Some thousands of end systems, world-wide, are currently managed by Operations Manager.

Operations Manager presents the status of the managed objects rather than the raw stream of events. This enables pro-active management of services. For example, defined thresholds can be set for filestore and swapspace which, when exceeded, will generate an alert automatically. This potential problem can then be detected and remedied without any disruption to crucial services.

Messages from the managed systems about the state of the managed resources are communicated via CAS. Icons representing services and service components change colour to indicate the significance and severity of a problem, or potential problem. Clicking on a service icon gives an instant view of the components underlying the service, their colour indicating their status. Thus the component at fault can be pinpointed swiftly. A service icon can map onto a specific service instance or the whole service, as required by the user.

By not relying on constant polling, the load on the IT network is kept to a minimum, freeing the maximum available capacity for business-critical applications. Also, alerts are reported immediately, and do not have to wait for the next poll.

Normally, each managed machine will be supported by an *Operations Agent* which reports status to one or more *Operations Manager*. When the manager establishes contact with the Agent, detailed information will pass from agent to manager about the current state of all the entities being monitored. From there on, only changes will be reported. Confidence in the report is maintained by a periodic 'heartbeat' message sent from agent to manager. If a beat is missed, the manager will know that the machine state is uncertain and it will re-establish control.

Remote Action facilities are provided by operations manager to give direct control of remotely located systems. Actions available depend upon the object type but for DRS/NX systems these include system resets, restarts and reload. The actions may be tailored to be any command relevant to the managed end system. The action messages and their replies are carried by CAS.

Operations manager uses both alerts and directed messages.

The operations manager agents for UNIX trap certain system and application specific events and raise alerts where necessary. The CAS message router directs these alerts to interested operations manager displays.

Most operations manager messages are of transient interest. Consequently 'discard on non-delivery' is requested for many of the messages. Non-delivery notification is not needed in most cases but is sometimes used to allow a re-establishment of confidence in the information flowing between manager and agent. For instance, all reports from agent to manager will be discarded on non-delivery, as they are likely to be out of date if not delivered promptly. The agent is notified about non-delivery of its 'heartbeat' and will suspend further communication with the manager. When the manager detects the loss of heartbeat, it can re-establish control of the agent.

4.2 OSMC Distribution Manager

OSMC Distribution Manager [Barthram, 1991] enables the centralised distribution of files, application software and products over a network. Files may be delivered to any kind of system supporting a suitable file transfer protocol. Software installation may be automated for UNIX, VME and PCs under Windows.

CAS is used to provide the messaging service upon which the *Distribution Manager* software depends. Messages are sent from the central *Management Application* software to remote agents controlling the transmission of files and the installation and activation of software on the remote systems. The remote agents transmit feedback information to the management application concerning progress and success or failure using CAS.

Unlike Operations Manager, Distribution Manager requires delivery of all its control messages. It does not use the discard on non-delivery capabilities.

4.3 Retail Systems Management

In 1984 ICL formed the Retail Business Centre to develop business in the vertical retail market [Pickworth, 1991].

This led to the development of products to allow retailers to make use of data collected at point of sale. Although many large retailers tend to use IBM mainframes at their head office, ICL decided against using IBM protocols for this purpose. This was for various reasons including the fact that the IBM standards were not open.

The systems management infrastructure components for management messaging (CAS) and bulk data transfer (CFT) were used instead to develop a set of open products known as *Retail Systems Management.*

These products are in use today by more than 100 retail customers worldwide.

4.4 Inland Revenue

The Inland Revenue (IR) currently uses ICL's Community Alert Management to carry management messages concerning their network of VME mainframes and BULL Office Systems. Each of the VME mainframes and over 1000 UNIX systems use the CAM protocol to communicate over the X.25 Government Data Network with a VME mainframe which acts as a central manager.

The CAM messaging system is used by several different management functions including:

- central statistics collection
- status monitoring system
- load set manager
- event logging system
- configuration management

Two kinds of message are used: alert messages and command messages. Alert messages carry information concerning an event or set of events which have occurred within a managed system. The alert messages have different classifications enabling the routing mechanisms to know what to do with them. Command messages are used to initiate actions on the managed systems. Command messages are also used to request information from the managed system, for example, the status of a managed object.

The *Event Logging Subsystem* is a significant user of CAM. This receives messages from a variety of sources concerning five general kinds of managed object:

- *X.25 Service*: which supports the management of the communications functions required to provide a network service conforming to the CCITT X.25 recommendations.
- *LAN*: which supports the management of all the hardware devices and communications functions required to provide a local area network service conforming to OSI standards.
- *Transport Service*: which supports the management of an OSI transport service over both the wide area and local networks.
- *File System*: which supports the management of the disk filestore on the UNIX systems.
- *Print Spooler*: which supports the management of print devices and the documents to be output on these devices.

Several thousands of messages per day are carried over this infrastructure within the IR.

5 Conclusions

This paper has shown how the technical architecture required for the management of a network of distributed system is a specialised form of client-server architecture. The features of ICL's management messaging service CAM have been described in detail. The CAM service works in real time and provides routing, resilience, recovery, store and forward and fan out features. Software implementing CAM is now available on a range of types of hardware and operating systems. This software forms an important part of ICL's TeamCARE™ systems management product set. It has also been used successfully by a range of ICL's major customers.

Trademarks

UNIX is a registered trademark of UNIX Systems Laboratories, Inc., in the U.S.A. and other countries.

X/OPEN is a trademark of the X/OPEN Company Limited in the U.K. and other countries.

ICL VME is a registered trademark of International Computers Limited in the U.K. and other countries.

OPEN*framework* and TeamCARE are trademarks of International Computers Limited.

References

BARTHRAM P., HOWLING D.; Distribution Management - ICL's Open Approach. *ICL Tech. J.* Vol 7, No 4, 1991.

BRENNER, J.B. Client-server architecture, *ICL Tech. J.* Vol. 9 Iss. 1 pp. 3-17, 1994.

GALE A.C.; An Evolution within ICL of an Architecture for Systems Management, *ICL Tech. J.* Vol 7, No 4, 1991

HACKER D.; Operations Management. *ICL Tech. J.* Vol 7, No 4, 1991.

PICKWORTH I.; Experience in Managing Data Flows in Distributed Computing in Retail Businesses. *ICL Tech. J.* Vol 7, No 4, 1991.

SMALL M. et al.; OSMC Operations Control Manager. *ICL Tech. J.* Vol 7, No 4, May 1991.

Biographies

Mike Small

Mike Small is Solutions Architect for ICL's Systems Management Business. In this role he has been responsible for the systems management solutions provided to some of ICL's major customers including France Telecom.

He is a graduate of Brunel University and joined ICT at West Gorton in 1967. He was initially involved in the design of engineering software for the design, manufacture and maintenance of ICL's 2900 series of computers and the ICL ME29.

In 1982 he was a founder member of ICL's Knowledge Engineering Group which was concerned with the exploitation of software technologies based on artificial intelligence. In this group he led the application of these technologies to systems management. He was responsible for ICL's VME Capacity Management System (VCMS) and received ICL's Gold Excellence Award for this work. He was also responsible for the development of the Operations Control Manager (OCM) which provided a graphical interface for the management of UNIX and VME systems.

He has published many papers in the ICL Technical Journal and the proceedings of international conferences.

Dave Roberts

Dave Roberts is Lead Designer for Operations Manager in ICL's Systems Management Business. He is responsible for the design, implementation and delivery of Operations Management solutions.

He is a graduate of Manchester University and has been involved in the computer industry since graduation in 1968.

Prior to his Systems Management involvement, he had wide experience in software development with both computer manufacturers and users. Over the past 20 years, he has taken part in several successful implementations of what we now understand to be client-server applications, including distributed data collection, remote data delivery, electronic mail and distributed transaction processing systems; hosted on ICL and other manufacturer's systems.

He has been with ICL since 1977; joined the Systems Management Business in 1985 and was one of the team which carried out the first CAM implementation on ICL's VME Operating System. He maintains a responsibility for the management messaging area, whilst majoring on the development of applications which exploit it. He is the recipient of an ICL Excellence Award for his work on Managed Bulk Data Transfer.

His current responsibilities include products which run on VME Mainframes, UNIX Servers and PC Clients.

PARIS - ICL's Problem & Resolution Information System

P.J. Loach
DSC, Customer Services, ICL, Manchester, UK

Abstract

This paper outlines the concept of PARIS (ICL's maintenance database), and some of the problems encountered during its development. The PARIS database was originally designed to record both hardware and software maintenance problems (although to date it is mainly software problems that have been recorded) by describing the symptoms by which incidents manifested themselves. Being a structured database this considerably simplified the task of identifying known problems and finding the appropriate corrective action, both for ICL and customer staff. This also paved the way for future automatic computer searching to take place, in a similar way to the operation of the Maintenance Knowledge Base (MKB). A further development allowing customer access to a subset of the data held on PARIS was the PC based CD-ROM system known as PC-PARIS.

1 Background

The introduction of the Series 39 family of computers, and the development of Support and Maintenance (SAM) in 1985, had shown that when a computer hardware fault occurred it was usually possible to obtain sufficient diagnostic information from the broken machine, which could then be transmitted to an ICL Mainframe in the form of a *fingerprint.* This fingerprint then could be compared automatically with previously collected sets to see if the problem was already known. In the event of a match being found an engineer could be dispatched to the site either with the appropriate spares, or knowing what action was required. In 1986 this method of diagnosis was enhanced to include some simple software problems relating to the firmware of the machine and the Maintenance Knowledge Base (MKB) was born. When a fingerprint match occurred then a solution (usually in the form of a repair) would be transmitted directly to the site. All this was achieved with a minimal amount of human intervention thereby saving time and costs for both customers and ICL. The principle of having to solve any problem manually only the first time it was encountered was, therefore, well established for both hardware and firmware. (For further information regarding the remote and automatic diagnosis facilities of Series 39 computers see [Allison, 1988]).

The resultant cost savings made by using fingerprints for hardware diagnosis, coupled with the increase in the reliability of computer hardware, meant that the next logical development was on the software front. Since the late 1970s ICL had been using a *loosely* structured database for the recording of *useful tips* and *known errors* called the Maintenance Data Base (MDB). A *field-trial* giving some of ICL's larger customers access to a subset of this data, via an ME29 computer, indicated that there was a demand for on-line access to this information.

The MDB, however, had been designed for internal use by ICL staff, and resided on an ICL Corporate Mainframe in Hitchin, (which contained other important ICL information). Also, being intended for internal use, much of the data recorded on it was unsuitable for customer viewing. The data was structured using *keywords* and therefore did not lend itself to an automated diagnosis, or search, process. To give all ICL customers access to the data recorded on the MDB, subsets were generated and reproduced on microfiche, commonly referred to as the Known Error Log (KEL), for distribution at fortnightly intervals. In the early 1980s it was realised by the support community that the massive increase in computer usage, and the development of ever more complex applications, would result in these microfiche becoming too cumbersome to be effective by the 1990s. Therefore it was decided in October 1984 to develop a new database based on the successful principle of the SAM/MKB fingerprints.

Shortly after this decision was taken, ICL started to develop another information system, designed to provide hardware assistance for engineers. This eventually grew into the LOCATOR system [Rouse, 1991]. The approach used by Locator was considered unsuitable for PARIS, with the result that the two systems were developed independently.

2 The Original Concept

Although ICL had the resources to design and build a suitable system, it was decided to employ outside contractors with specialist knowledge in the use of databases. As some of them had been involved in the original design of the MDB, they were aware of the nature of the data that needed to be recorded on PARIS.

They advised that a fingerprint approach should be adopted, whereby the symptoms of a problem identify it uniquely. The ICL Integrated Data Management System (IDMS), first released in 1977, was recommended as the best method of recording and searching through large volumes of data efficiently. An added advantage was that the IDMS database system readily lent itself to expansion, as required, to include new products.

It was decided to mirror, as far as possible, the data structure on the existing product hierarchy, with the topmost node being GLOBAL, having one leg for hardware and another for software. (The structure consists of a series of nodes, each dependant on the one above and, as you progress down the structure, each node becomes more product specific.)

It is interesting to note that when several groups of diagnosticians were asked independently to invent a suitable structure, each group made a completely different suggestion.

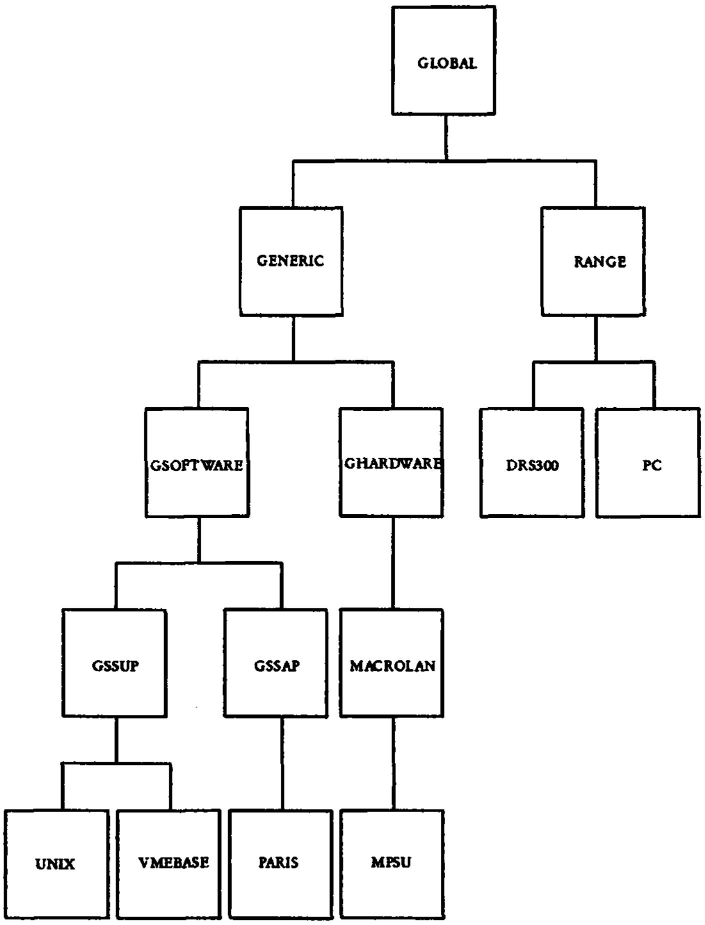


Figure 1 Extract from the PARIS hierarchy

Each product could then individually generate its own *bespoke* set of Support Knowledge Entries (SKEs) and Symptoms. (A SKE is a record of the manifestation of a problem and consists of a global *Entry Class* and a series of local product related *Symptoms*, together with a brief textual description of the way the problem appears to the user.) The global Entry Class was intended to provide a description of the type of problem (e.g. unexpected Result Code or Program Error) which would be applicable to all products and provision for 9998 such entries was made (of which only 41 have been used to date). The local Symptoms and associated values would further describe the problem in terminology most appropriate to the product. These SKEs would, wherever possible, be linked to records containing solutions to the problems. The result was a highly structured database as opposed to the existing MDB.

An example of a typical SKE is:

SKE NUMBER:	60035
SKE TITLE:	SETPERMISSIONCONTROL fails with RC 9118
ENTRY CLASS:	0015
SYMPTOMS:	
PRODVR	SV292
RC	9118
ACTVTY	DECLARATION
COMAND	SETPERMISSIONCONTROL
OBJTYP	PERMISSION
PRODCT	CATLOG
OBJTYP	OPTION
DESCRIPTION:	

SETPERMISSIONCONTROL fails with RC 9118 when it should be generally available.

3 Populating the Database (part 1)

As customers were intended to be given access to PARIS data it was decided that the Product Authorities (i.e. those responsible for the development and support of products) should ultimately be responsible for all the data on PARIS, so a very secure privacy/authorisation system was set up to *vet* the quality of the data recorded. ICL's flagship product (VME) was selected to test the PARIS concept and the VME Product Authority was tasked with generating and entering the first SKE standards. This resulted in an empty database shell-like structure with a set of SKE standards on it. The task of populating it was given to the diagnosticians working in the Software Support Centres (SSCs) in the course of their normal work, investigating incidents. There was an expensive learning curve for these diagnosticians as, in addition to their normal work, they had to record information (SKEs) as they encountered new problems. Naturally there was considerable resistance to this, since recording this data did not assist them in the diagnosis process and involved them in an additional 20 minutes extra work per problem. The result was that fewer than 500 SKEs were created during the first six months, and closer inspection showed that some were of very dubious value.

Throughout this six month period there had been a succession of meetings to monitor the process and consider how it could be improved. This resulted in a *Change Control Board* being set up, consisting of some of the PARIS designers, the Product Authority and SSC diagnosticians, to advise and progress any alterations to PARIS. At the end of the six month period two issues had become clear:

1. Although the basic concept of PARIS was sound, the Product Authority had failed to generate a set of SKE standards suitable for use by the SSCs.
2. Any future trial would need to have a pre-populated database otherwise no-one would use it.

4 Populating the Database (part 2)

The VME Product Authority, in consultation with members of the SSC, embarked on the task of re-writing the SKE standards. At the same time the PARIS development team was tasked with producing a programme to convert the data held on the MDB to a format suitable for inclusion on PARIS. One of the problems encountered in the conversion process was that some of the MDB data either could not be converted automatically, or that its conversion resulted in multiple SKEs being produced. Both these situations could potentially have resulted in the loss of existing knowledge. A suite of transition programmes was written to overcome most of these difficulties and many MDB entries were re-written as 'SUPERCEDED by...' (it really was mis-spelt!) to prevent them from being accidentally updated without the equivalent SKE on PARIS being changed. Throughout the entire development period it was considered essential that any data entered on PARIS should automatically be entered on the MDB, so the data would be available to customers on the KEL microfiche. To achieve this, a programme was run daily to copy any new, or altered, data from PARIS to the MDB. A further trial was held using the pre-populated database with more encouraging results, and occasionally problems were actually identified and solved using PARIS alone.

Statistical evidence indicated a problem for SSC staff in identifying existing PARIS SKEs where the original SKE had been raised by a member of a Product Support Unit. An experiment was conducted in which ten randomly selected problems, where it was known that a SKE had been raised by an SSC diagnostician, were given to other SSC diagnosticians. The result was a 100% success rate in identifying the original SKE raised.

A similar experiment where the SKE had been raised by a Product Authority diagnostician resulted in only a 40% success rate. The conclusion was that the way an incident is described in a SKE depends on the level of knowledge of the person raising the SKE. It appeared that the higher this knowledge was, then the more likely that the SKE would explain the reason for the problem rather than the manifestation of the fault. This resulted in the rule that all problems arriving at the SSC should have a SKE raised, which could then either be modified or another SKE raised and linked to the original SKE by the Product Authority if necessary.

5 A Further Development

At about this time the development team suggested that an improvement to the basic PARIS concept would be to have a SKE (or a number of SKEs) pointing to a Problem which could then have a Solution (or several solutions) linked with Solution Actions. This increased the flexibility of the data being recorded and resulted in the current design structure for PARIS. The difficulty with introducing this change was that previously the SKE had held both the manifestation of the problem and a brief description of the problem itself. Although the definition of a SKE was changed to accommodate this introduction of a Problem record, there was (and still is) some confusion as to where the description of the manifestation ends, and the description of the problem begins.

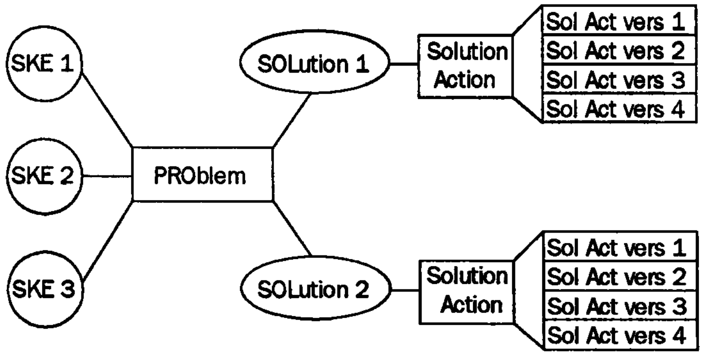


Figure 2 PARIS linking of SKEs to Solution Actions

The figure illustrates a typical arrangement where one problem manifests itself in three different ways (3 SKEs). There are two Solutions (one might be advice on avoiding the problem, and the other the cure by way of a repair or patch). Both these solutions have a Solution Action having four different Solution Action Versions, (the applicable version would depend on which version of the product is giving rise to the problem).

6 Acceptance

By 1988 the success of PARIS as ICL's future support knowledge database looked secure and it was decided to stop updating the MDB. The complications of maintaining two databases with so much duplicated data were considerable, so it was decided to transfer all the remaining known useful data onto PARIS by 1st April 1989, for all products. With the anticipated demise of the MDB, and the KEL microfiche, a working party was set up to advise on the format of a PARIS microfiche. During 1988 it became clear that a large quantity of useful VME data on the MDB could not be converted automatically, and many other products would need to re-write their data totally. This resulted in the MDB getting several reprieves. By 1991 ICL was still in the unenviable position of supporting two massive databases containing almost identical information, but in completely different formats. As both databases now had text search facilities using Content Addressable Filestore (CAFS - an ICL product developed in the early 1970s enabling very fast parallel searches), and this was the normal process used to retrieve information in SSCs, a deadline of 31st December 1992 was set (and kept) finally to switch off the MDB.

7 PC-PARIS

Throughout the development of PARIS it was anticipated that customers would be charged a fee for *on-line* access to PARIS data. Various ideas were suggested ranging from the use of British Telecom's 0898 premium rate service numbers, to charging for a licence to access data. Both of these methods would have required an expensive infrastructure of smaller computers to *front-end* the service and poor response times could have resulted if the service had become over-subscribed. The rapid growth of the PC market and the development of CD-ROMs made it possible to consider these as a vehicle for distributing the data.

Rather than develop new software, it was decided to adopt an existing commercial package Windows Personal Librarian which had sophisticated text searching facilities and allowed the *encryption* of the data so it could be password protected. The CD-ROM's capacity is about 600 MBytes enabling the PARIS data to be grouped into a number of discrete files, each protected by a different password. The customer could then choose the appropriate file(s) for his business and, by paying the licence fee, would receive a new CD-ROM and password(s) at regular intervals. The adoption of this mode of access to PARIS data also satisfied the demands of secure sites and those where no external communication facilities existed.

Although most ICL staff have access to the Mainframe running the PARIS service, there are occasions when this is either expensive, or undesirable. Also, multiple attempts to access the same data can result in either poor response times or even *deadlock*. To ensure that the data on the CD-ROM is up-to-date, the customer disc is produced every two weeks, and a disc for ICL use is generated in the intervening weeks. (The ICL internal disc has additional engineering data on it which is not relevant to customers.)

8 The Future

As with any project of this magnitude, there are problems that were either forgotten or considered to be minor at the time of the original development. One of these is that no provision was ever made to archive or delete data from PARIS for products that were no longer supported. One result is that, as the database expands, space is becoming a problem especially on the internal ICL CD-ROM. Also, customers are starting to encounter a significant volume of data which is no longer of any value to them, and this masks the more recent information which may be relevant. This issue is being addressed and a further programme is being developed to assist the Product Authorities to remove obsolete SKEs.

As stated previously, there is virtually no hardware data recorded on PARIS, although investigations into how to organise the hardware SKE standards have started. Originally the data on PARIS comprised a list of faults on mainframe systems. This data has now been expanded to include information pertaining to the small and medium range systems, and it has been broadened to include Advice & Guidance SKEs to help users exploit ICL products (especially for Office Systems products).

9 Some Facts and Figures

Currently the Mainframe PARIS database uses 1.2 GBytes and runs on a 3-node Level 80 machine. There are about 60,000 SKEs, 31,000 Problems and 30,000 Solutions with 31,000 Solution Actions. There have been about 600 Change Requests made by the PARIS users, and discussed by the Change Control Board since the beginning, of which around 350 have been approved and incorporated in the product.

The PARIS user community is world-wide, consisting of over 3,000 registered users accessing the Mainframe service. In addition, on alternate weeks, 500 CD-ROMs are distributed to ICL support staff, with a further 400 CD-ROMs being produced for external licensed customers.

Average search times are 50 seconds on Mainframe PARIS (this does not include any time associated with the comms), and 15 seconds is typical for a complete search of all the SKEs on the PC-PARIS (ICL disc) using a 486 PC.

Acknowledgements

Windows Personal Librarian is a trademark of Personal Library Software Inc.

Thanks to all my colleagues who have contributed, knowingly or unknowingly, to this paper.

Glossary of Abbreviations and Terms used

CAFS	-	Content Addressable Filestore
CD-ROM	-	Compact Disc - Read only Memory
firmware	-	basic software built into the machine
IDMS	-	Integrated Data Management System
KEL	-	Known Error Log
MDB	-	Maintenance Data Base
MKB	-	Maintenance Knowledge Base
PARIS	-	Problem and Resolution Information System
PC-PARIS	-	Version of PARIS designed to run on PCs
PROblem	-	The cause of the manifestation detailed in the Support Knowledge Entry
SAM	-	Support and Maintenance
SKE	-	Support Knowledge Entry
SOLution	-	How to resolve the Problem
Solution Action	-	The activity required to implement the Solution
Solution Action Version	-	The specific Version of any repair required.
SSC	-	Software Support Centre
VME	-	Virtual Machine Environment

Terms such as GSAPP, GSOFTWARE and GSUPP used in figure 1 are examples of names given to various nodes in the hierarchy. In this particular instance they refer to Generic Software APPlications, Generic SOFTWARE and Generic Software SUPervisor respectively.

References

ALLISON, R., ICL Series 39 Support Process, *ICL Tech. J.* Vol. 6 No 1 pp2-16, 1988.

ROUSE, G.W., *LOCATOR:* An Application of Knowledge Engineering to ICL Customer Service, *ICL Tech. J.* Vol. 7 No 3 pp546-553, 1991.

Biography

P.J. Loach
Peter Loach has worked for ICL for 28 years, having spent the first 11 years as a field based hardware engineer, followed by 17 years working, at first in the Liverpool and later in the Manchester Software Support Centres. Since 1986, alongside his work in the Manchester Support Centre, he has worked closely with the PARIS development team and has been involved in the development of the product and the user interfaces of both PARIS and PC-PARIS. His current role as the Service Controller includes Chairing the Change Control Board, and he acts as the focal point for all the user problems etc., with the PARIS suite of products.

Architecture Briefings

A Briefing is a short paper on a technical issue. ICL produces a number of valuable, informative, leading-edge papers each written by an authority in a particular field.

The Briefings are published quarterly, in sets of 5, at a cost of £250 per annum to OPEN*framework* Community members and £350 to ICL customers.

The 1994 current titles include:

94/01	Object Management Group
94/02	X/Open - The Open Systems Integrator
94/03	Electronic Mail - A New Era in Interoperability
94/04	Client-server Middleware: The Way Ahead
94/05	Object Database Technology - How Mature Is It?

1993 back issues:

93/001	SQL Interoperability - The New Standards
93/002	CALS
93/003	Classification of Performance Problems
93/004	What is Client-server Computing?
93/005	Business Trends Related to Transaction Management
93/006	Repositories: Is the Dream Fading?
93/007	Systems Management - Many Things to Many People
93/008	ATM: The New Networking Technology
93/009	Open Transaction Management - Standards Update

For further information and details on how to order, please contact the ICL SystemWise Help Desk, D2D House, Manchester Road, Ashton-under-Lyne, Lancashire, OL7 0ES, England.

Telephone +44 (0)81 565 7993 Fax: +44(0)61 371 9164

Ingenuity

The ICL Technical Journal

Guidance for Authors

1 Content

Ingenuity, the ICL Technical Journal, has an international circulation. It publishes high standard papers that have some relevance to ICL's business. It is aimed at the general technical community and in particular at ICL's users and customers. It is intended for readers who have an interest in the information technology field in general but who may not be informed on the aspect covered by a particular paper. To be acceptable, papers on more specialised aspects of design or application must include some suitable introductory material or reference.

Ingenuity will usually not reprint papers already published but this does not necessarily exclude papers presented at conferences. It is not necessary for the material to be entirely new or original. Papers will not reveal matter relating to unannounced products of any of the ICL Group companies.

Letters to the Editor and book reviews may also be published.

2 Authors

Within the framework defined in paragraph 1, the Editor will be happy to consider a paper by any author or group of authors, whether or not an employee of a company in the ICL Group. All papers are judged on their merit, irrespective of origin.

3 Length

There is no fixed upper or lower limit, but a useful working range is 4000-6000 words; it may be difficult to accommodate a long paper in a particular issue. Authors should always keep brevity in mind but should not sacrifice necessary fullness of explanation to this.

4 Abstract

All papers should have an Abstract of not more than 200 words, suitable for the various abstracting journals to use without alteration.

5 Presentation

5.1 Printed (typed) copy

Two copies of the manuscript, typed 1½/2 line spacing on one side only of A4 paper, with right and left margins of at least 2.5cms, and the pages numbered in sequence, should be sent to the Editor. Particular care should be taken to ensure that mathematical symbols and expressions, and any special characters such as Greek letters, are clear. Any detailed mathematical treatment should be put in an Appendix so that only essential results need be referred to in the text.

5.2 Disk version

Authors are requested to submit a magnetic disk version of their copy in addition to the manuscript. All artwork and diagrams to be supplied in their original source format. The Editor will be glad to provide detailed advice on the format of the text on the disk.

5.3 Diagrams
Line diagrams will, if necessary, be redrawn and professionally lettered for publication, so it is essential that they are clear. Axes of graphs should be labelled with the relevant variables and, where this is desirable, marked off with their values. All diagrams should have a caption and be numbered for reference in the text and the text marked to show where each should be placed - e.g. "Figure 5 here". Authors should check that all diagrams are actually referred to in the text and that all diagrams referred to are supplied. Since diagrams are always separated from their text in the production process, these should be presented each on a separate sheet and, *most important*, each sheet must carry the author's name and the title of the paper. The diagram captions and numbers should be listed on a separate sheet which also should give the author's name and the title of the paper.

5.4 Tables
As with diagrams, these should all be given captions and reference numbers; adequate row and column headings should be given, also the relevant units for all the quantities tabulated.

5.5 References
Authors are asked to use the Author/Date system, in which the author(s) and the date of the publication are given in the text, and all the references are listed in alphabetical order of author at the end.

e.g. in the text: "...further details are given in [Henderson, 1986]"

with the corresponding entry in the reference list:

HENDERSON, P. Functional Programming Formal Specification and Rapid Prototyping. IEEE Trans. on Software Engineering SE*12, 2, 241-250, 1986.

Where there are more than two authors it is usual to give the text reference as "[X et al ...]".

Authors should check that all text references are listed; references to works not quoted in the text should be listed under a heading such as *Bibliography* or *Further reading*.

5.6 Style
A note is available from the Editor summarising the main points of style - punctuation, spelling, use of initials and acronyms etc. - preferred for Journal papers.

6 Referees
The Editor may refer papers to independent referees for comment. If the referee recommends revisions to the draft, the author will be asked to make those revisions. Referees are anonymous. Minor editorial corrections, as for example to conform to the **Ingenuity** general style for spelling or notation, will be made by the Editor.

7 Proofs, Offprints
Printed proofs are sent to authors for correction before publication. Authors receive up to 20 offprints of their papers, on request, free of charge, and further copies can be purchased.

8 Copyright
Copyright of papers published in **Ingenuity** rests with ICL unless specifically agreed otherwise before publication. Publications may be reproduced with the Editor's permission, which will normally be granted, and with due acknowledgement.

Ingenuity May 1994