DSTO

Distributed Computing Environment: An Architecture for Supporting Change?

J. Mansfield and J. Clothier

DEPARTMENT OF DEFENCE
DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION

Distributed Computing Environment: An Architecture for Supporting Change?

J. Mansfield and J. Clothier

Information Technology Division
Electronics and Surveillance Research Laboratory

DSTO-CR-0007

ABSTRACT

Distributed Computing Environment (DCE) has been in development for about five years but has only been widely used in the last two years. It consists of a number of services that have been selected from current proprietary distributed facilities and integrated so that they work together. Together these services form an architecture for distributed computing that enables users to carry out the new, cheaper operations they require with the interoperability, reliability and security standards of mainframe computers.

The facilities provided by DCE are often reviewed from either the perspective of reliability or the perspective of interoperability. This paper reviews the facilities of DCE from the perspective of change. An architecture is proposed which will support the evolution of information systems.

Published by

DSTO Electronics and Surveillance Research Laboratory
PO 1500
Salisbury. South Australia, Australia 5108

Telephone: (08) 259 7053
Fax: (08) 259 5619

AR-009-456
November 1995

APPROVED FOR PUBLIC RELEASE

Distributed Computing Environment: An Architecture for Supporting Change?

EXECUTIVE SUMMARY

Over the past twenty years information systems architectures have migrated away from the centrally located mainframe in favour of geographically distributed services. The disadvantages of a distributed system are poorer reliability, interoperability and security. Distributed Computing Environment (DCE) is a software standard and technology which provides a very high level of reliability, interoperability and security within distributed information systems.

DCE has been in development for about five years but has only been widely used in the last two years. It consists of a number of services that have been selected from current proprietary facilities and integrated so that they work together. DCE is widely recognised as a tool for providing interoperability between heterogenous computer systems. It has the potential to provide computer systems which can more readily adapt to change.

There are three basic elements of a computer program: the user interface, the business rules and the data. As programs have increased in size and complexity, tools and standards have emerged to support these three elements as separate entities.

If the separate elements of a computer program are executed on different computers then elements must make requests to other elements for services. The recipient computer executes the requested service and sends the response back to the requesting computer. Computers requesting a service are known as clients. Computers providing a service are known as servers. It is even possible for a computer to act as both a client and a server. Mapping the three element view of a computer program to a client/server architecture, such as DCE, requires a decision to be made on which elements are clients and which elements are servers.

A three layer client/server architecture may have the business rules vested in the client, with servers supporting the interface and database (function) services. The clients are created by people who are familiar with the organisational or business procedures whilst the expertise used to create the servers is that of technical experts. By breaking the system down into parts the ramifications of change are minimised as each part may be changed individually.

The three layer client/server architecture becomes difficult to visualise when some components act as both clients and as servers. However it can be viewed as three concentric rings if the servers are combined into one layer and the physical world becomes the outer ring. A concentric view highlights the potential of the three layer architecture to support change. The concentric architecture can then be extended to view the components as lots of little "machines" that have input and output and process data. With this view they can be replicated and dynamically combined in various ways to suit the needs of the user.

The essence of minimising the ramifications of change is that the granularity of the component logic is very small. Although this leads to high overheads they may be balanced, if not outweighed, by the benefits.

Authors

John Mansfield
Information Technology Division

John Mansfield is a Senior Professional Officer with DSTO's Information Technology Division. He is the leader of a section responsible for research into command support system architectures with particular interest in distributed systems. John has thirty years experience in engineering and computer system development for business, academe and government.

Jennie Clothier
Information Technology Division

Dr Jennie Clothier is a Senior Research Scientist with DSTO's Information Technology Division. She is the leader of a section responsible for researching and developing decision support systems. Before joining DSTO in 1990 Jennie worked for the UK Royal Navy, researching the application of information technology to tactical command and control.

Contents

Glossary

CORBA	Common Object Request Broker Architecture.
DBMS	DataBase Management System.
DCE	Open Software Foundation's Distributed Computing. Environment. An example of middleware.
DOS	Microsoft Disk Operating System.
GDA	Global Directory Agent.
IDL	Interface Definition Language.
Kerberos	A security facility originating from MIT's Athena project.
LAN	Local Area Network.
middleware	Software that lies between the operating system and application software. Its purpose is to facilitate communication between systems.
OS/2	An IBM operating system for Intel based computers.
POSIX	Portable Operating Systems Interface, an IEEE standard. originally intended to standardise Unix systems; now extended to IBM MVS, DEC VMS, etc.
Stub	Program that interfaces the component to the communications environment.
UUID	Universal unique identifier.
Unix	A non-proprietry operating system supported by many hardware vendors.
WAN	Wide Area Network.
X.500	An OSI directory standard.

1. Introduction

The information systems technology of today is in a constant state of change due to the pace of invention and innovation in the computer and communications industries. Organisations are also in a state of change; either evolutionary change to meet the needs of the market place by increments or revolutionary change by business re-engineering. A major problem is the reconciliation of these two trends with the need to continue to do business in a cost effective way.

None of these problems is new, what is new is the pace of change. Twenty years ago users were frustrated by a huge backlog of software applications which they needed to be developed on their mainframes. This problem has not inherently changed, even though the pace at which applications are developed has increased enormously. The problems of moving from one client / server based application to the next are the same as moving from one mainframe application to the next. Whilst lessons may have been learnt from the old application, and hopefully applied to the new application, most of the investment in development, training, and procedures is thrown away.

What is needed is an architecture which will respond to change in an evolutionary way, building on what has gone before. The requirements for this architecture are:

- the architecture must be simple in structure,
- the systems built within the architecture must be easy to change,
- there should be no wholesale recompilation when an application is changed - in the ideal case a user makes the change.

Five to eight years ago the majority of computer systems were mainframes with either dumb or smart terminals attached to them. These terminals could be local or remote but they could only communicate with the mainframe. If a user required to send a message to another user on a different machine the message was sent mainframe to mainframe over a communications link with a modem at each end. The major benefits of this system were easy access to large databases, high computing power and a high level of security.

As workstations and PCs gained in power it became obvious that it would be simpler and considerably cheaper for the majority of tasks to be processed within the local machine. These machines still had to communicate with each other and with mainframes to gain access to data held on other machines so the local area network (LAN) was developed. The LAN provided high data rate information exchange between local machines but still used the older mechanism for communication between LANs. The wide area network (WAN) improved this communication mechanism so that it appeared to the user that they were using one machine when in fact they may have been using data, programs and processing power from a number of machines. Today a workstation on a LAN/WAN provides many of the facilities of a mainframe such as access to large databases and high computing power.

The disadvantages of these distributed systems are that they are less reliable and have poorer interoperability and security than a mainframe. To bring the geographically distributed system back to the standards accepted by mainframe users it is necessary to create a standard that lies between the applications the users wish to run and the numerous operating systems on the various proprietary platforms. A number of standards for this "middleware" have been proposed but the Open Software Foundation's Distributed Computing Environment (DCE) is the only one that has been widely adopted by the vendors of computing hardware.

DCE has been in development for about five years but has only been widely used in the last two years. It consists of a number of services that have been selected from current proprietary distributed facilities and integrated so that they work together. Together these services form a philosophy for distributed computing that enables users to carry out the new and cheaper operations they require with the interoperability, reliability and security standards of mainframes.

The facilities provided by DCE are often reviewed from either the perspective of reliability or the perspective of interoperability. This paper reviews the facilities of DCE from the perspective of change. An architecture is proposed which will support the evolution of information systems.

2. Background

2.1 The history of DCE

Distributed computing is not new. Several thousands of sites around the world claim to be running distributed computing services. However, these implementations have few relevant standards, are based on proprietary solutions, and only offer partial interoperability solutions, relying on the expertise of developers to patch together systems.

The main challenges of any distributed system are summarised below.

- Interoperability, the ability of two systems to exchange information in a standard form. This may extend to users of one platform running programs on another platform and even applications being distributed across multiple computers.
- Consistent support for heterogenous computers from multiple vendors.
- Identification of distributed services and resources.
- Provision of good security.
- Support for end users, application developers and system administrators.
- High availability and good performance.

DCE is an interoperability standard for open distributed computing. It was developed by the Open Software Foundation (OSF) which is a consortium of over 360 members including commercial, government, and university groups. OSF was originally a technology integrator and distributor of Open Systems Technology with over 300 employees worldwide. Its technology products have included an operating system (OSF/1 Release 1.2) , a visual user interface and toolkit (Motif 1.2) and a distributed computing environment (DCE 1.0.2A). Today, OSF no longer develops products. Its main role is to oversee the development of OSF products by industry.

OSF issued its request for DCE technology in 1989. A year later, after several submissions and reviews, an OSF DCE technology was selected. It then took another two years for OSF to deliver version 1.0 of its DCE product. The first OSF DCE vendor products began to emerge in late 1992 to mid 1993, but these versions suffered teething problems and reliable OSF DCE implementations were not widely available until late 1993 to early 1994.

2.2 Basic services needed by distributed systems.

Computer programs or rather applications which execute on a single machine are known as centrally run programs. In the 1970's and 1980's most programs were like this. The program was executed on a central mainframe computer. A number of terminals were attached to the mainframe and these acted as simple display devices which carried out no processing themselves.

Centrally executing mainframe programs are very efficient. Control overheads are low when management of a program is centrally administered. However, organisations that rely on such computer services are vulnerable because if the central computer fails, all operations are lost. This recently happened to the Commonwealth Bank where all automatic teller machines and electronic counter transactions were lost for half a day due to the failure of a single computer.

One way of overcoming the vulnerability of centrally executing mainframe programs is to run a duplicate machine. But mainframe computers are exceptionally expensive and there is no guarantee that both systems will not fail at the same time.

The alternative to central execution is distributed execution. Distributed execution is when the program or application is executed using a number of different computers. In these cases the application must make a request to another computer for a service. The recipient computer then executes the requested service and sends the response back to the requesting computer. Computers requesting a service are known as clients. Computers providing a service are known as servers. It is possible for a computer to act as both a client and a server (see Figure 1).

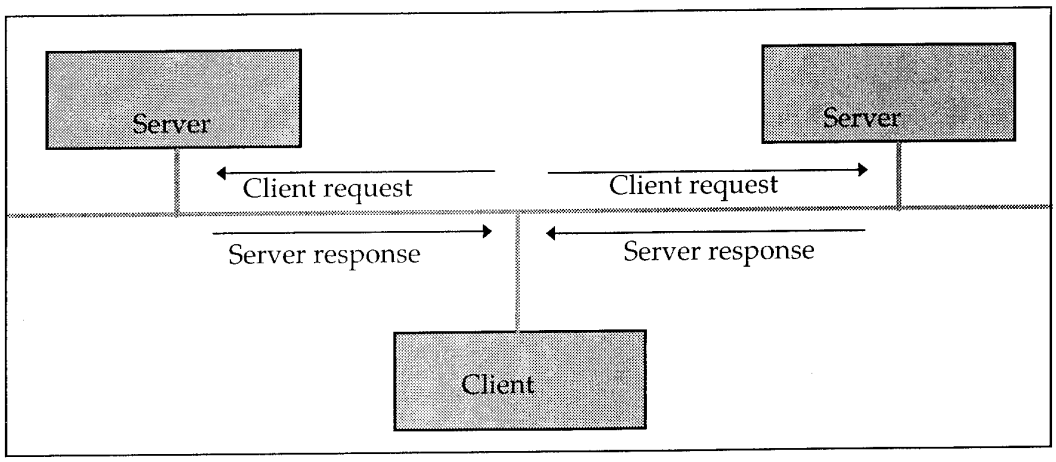


Figure 1. A very simple client/server architecture. A single application uses the services of several computers. This requires the part of the application hosted by the client computer to request services from the another part of the application hosted on the server computer.

As soon as an application becomes distributed then there are a number of facilities which become critical to its smooth execution.. Some of these facilities are necessary to give the illusion that the application is running on a single machine, others such as time and security servers perform functions that are unnecessary when the application was running is running on a single machine. Figure 2 provides an illustration of the facilities provided by the Open Software Foundation's DCE. Each facility is briefly described below.

- **Time.** In a distributed computing environment it is important that the computer clocks are synchronised, otherwise an application may show unpredictable behaviour. Without this synchronism there could be chaos as locally each

machine may have a different system time and some machines may even be in different time zones. For the purposes of file updating, transaction processing, security, backing up, compilation, etc. it is crucial that all machines have the same reference time.

- **Naming.** When multiple computers are used it is important to provide a universally consistent way of naming programs and people. Inconsistent naming could lead to either the wrong service being provided or a clash of services.
- **Remote Procedure Calls.** Procedure calls take on new meaning in a distributed computing environment. A computer program normally consists of a set of procedures which execute sequentially. A procedure may request that a file be opened and data read from it. However, it is always assumed that the file and the procedure for opening it exist on the same machine. When the file and the procedure for opening that file exist on another (remote) machine, then the requesting program must make a call to that remote procedure. This is known as a remote procedure call and involves knowledge of the communication systems used to transmit the data.
- **Threads.** A program which makes procedure calls to other computers obviously has the potential to slow down the rate at which an application runs whilst it waits for a response from the other computer. Threads are a popular way of improving application performance. They introduce parallelism and have been used in various versions of UNIX and OS/2 for some years. A procedure may be duplicated within the program to form threads. These threads execute independently but share static and external data. A server program using threads could handle multiple client requests simultaneously and thus minimise delays within the system.
- **File System.** Different computers have different file systems. For example, an MSDOS file system is very different to a UNIX file system. To overcome this problem in a distributed environment there is a file service that gives the appearance of a single file system which incorporates all the files available within the system. So users and computers can use this service to easily share files and the information in those files.

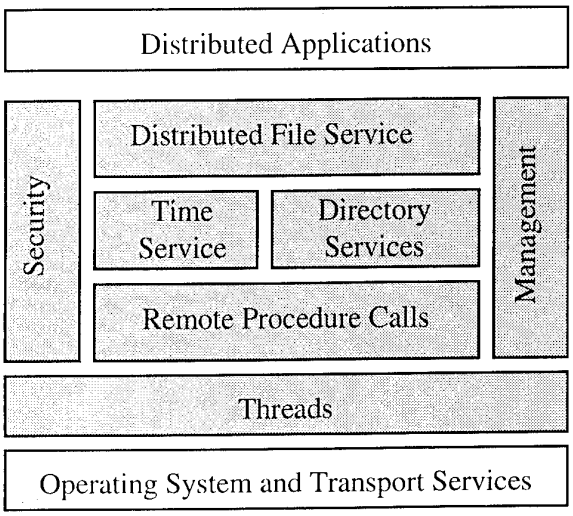


Figure 2. The services provided by OSF's DCE are represented by shaded boxes. They include: a Distributed File Service, a Time Service, Directory Services, Remote Procedure Calls, Threads, Security and Administration.

- **Security.** Protection of computer resources, such as files and applications, from unauthorised access requires a user to be regarded as authentic and to have the necessary authority. This usually requires a user to enter a password as proof that he or she is who they claim to be. Access to files is then controlled through permissions or privileges associated with each resource. When the number of users is small, a single computer may manage all of the passwords and permission functions. For a distributed computer system, there may be a very large number of users accessing an even larger number of resources. It would be impractical to maintain every user's security information on every computer in the system. The answer lies in the provision of a single centralised database which serves security information.
- **Management.** A distributed computer system has a much higher management and administration overhead than a centralised computer system. Tools for administrative support which automate some of these routine tasks help to reduce the management burden of distributed systems. For administrative and operational purposes distributed services are collated into a basic unit which is referred to under DCE as a cell. A cell is a group of users, systems and resources that typically have a common purpose and share common services. At a minimum, a cell consists of a directory (name) service, a security service and a time service. Usually a cell consists of nodes in a common geographic area, but geography does not necessarily determine its boundaries. Boundaries of a cell, in terms of the number of systems and users, are influenced by four basic considerations: purpose, administration, security and overhead. Figure 3 shows the services provided to a cell in a distributed system.

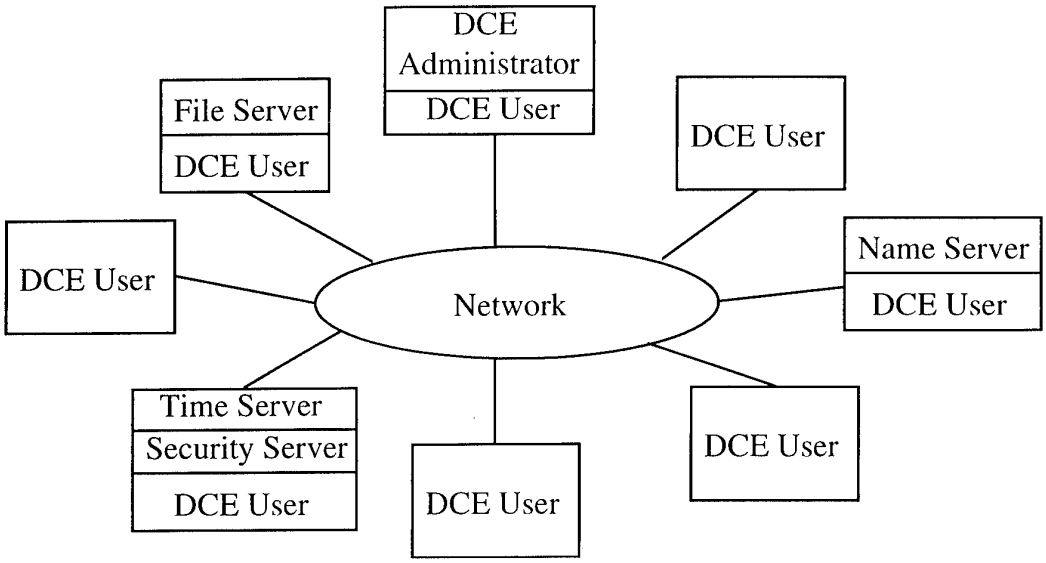


Figure 3. A DCE cell is an independent set of clients and servers, managed as a group. Cells can be combined to form multi cell systems.

A more detailed description of DCE's services is given in Appendix A. Appendix B provides a short description of the Defence Science and Technology Organisation's experience in establishing a DCE cell.

3. A Logical View of Computer Programs

3.1 A three level architecture

A computer program has three basic elements. There is the data, the logic and the user interface. Simple programs combine all three elements. For example, a program normally declares the acceptable data types, states the procedures to be executed (and the sequence in which they are to run) and writes output or takes input from some other device, such as a terminal (see Figure 4).

As programs have grown in size and complexity, attempts have been made to provide tools dedicated to the provision of data, the logic of the program and the user interface. For example, databases have been designed to store data which exists beyond the life of the program's execution, and user interface builders have been developed to allow the easy development of panes, menus and windows.

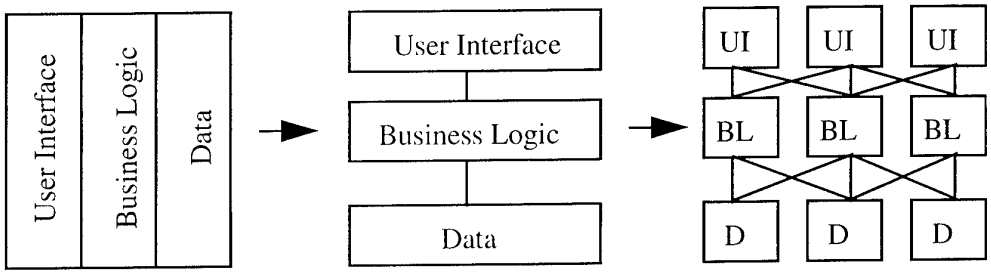


Figure 4. There are three basic elements of a computer program: the user interface (UI), the business logic (BL) and the data (D). As programs have increased in size and complexity, tools and standards have emerged to support these three elements as separate entities. It is possible that a number of highly networked elements may constitute an application.

The three level architecture illustrates the state of the art. More flexibility could be achieved if the elements of the current three level architecture were broken down into discrete components.

3.2 Mapping computer programs to a distributed architecture

Mapping the three layer logical view of a computer program to a client/server architecture, such as DCE, requires a decision to be made on which elements are clients and which elements are servers. A three layer mapping is given in Figure 5.

- The top layer is the interface to the user and to other systems; it utilises specialised servers, such as Motif, to provide an attractive and useful interface to the user.
- The middle layer contains the organisational logic. It consists of business rules grouped into modules called clients. Clients accept information from the interface servers, process it and return the information to the interface servers.
- The bottom layer provides functions such as data storage and printing that are common to many clients and hence are extracted into function servers.

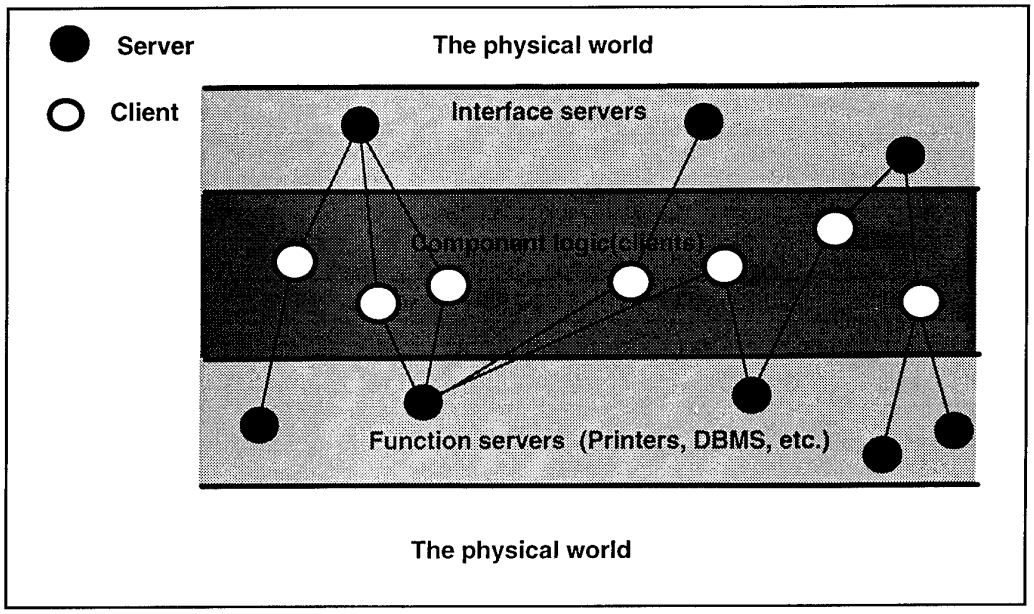


Figure 5. A three layer client/server architecture

The benefits of the three layer architecture are two fold:

1.The clients are created by people who are familiar with the organisational or business procedures whilst the expertise used to create the servers is that of technical experts; database designers, human computer interface designers, and windows programmers. The servers can be used by any client that needs them and, indeed, the clients can become "servers" to other clients if they contain commonly used procedures. The purpose of an information technology system is to aid the organisation to become more efficient and more effective. By applying particular expertise only where it is needed the system becomes more flexible and quicker and cheaper to implement.

2.By breaking the system down into parts the effect of making a change is minimised as only the part, or parts, involved need be changed. If the database manager is changed from one vendor to another, the only changes required are in the database access server. Provided the database interface remains unchanged the clients, and the users, can remain unaware of the change.

4. DCE as a Technology to Support Change

"Notice we didn't mention rewriting the application: redesigning and rewriting are constant. Why? Because the group of somewhat interdependent clients and servers across the network probably won't be thrown away at the same time....."

From Understanding DCE
Rosenberry, Kenney & Fisher 1992

DCE is widely recognised as a tool for providing interoperability between heterogenous computer systems. It also has the potential to provide computer systems which can more readily adapt to change. These changes may be the transition of an application from one computer to many or perhaps changes within an application's clients or servers. A change in an user interface server, in the business rules in a client or replacement of a DBMS server.

4.1 A concentric ring architecture

The three layer client/server architecture becomes difficult to visualise when some components act as both clients and as servers. However it can be viewed as three concentric rings if the servers are combined into one layer and the physical world becomes the outer ring. A concentric view highlights the potential of the three layer architecture to support change. Figure 6 shows a concentric representation.

In the outer ring lies the physical world, in the middle ring the service providers and the centre the business rules. It can now be seen that the clients and servers are each a special case of a software component and the three layer architecture breaks down into a multi-level architecture. Thus a better view may be a single ring (Figure 7), inside the ring is an abstract structure and outside the ring is the physical world. In accordance with current terminology the program for the business rules will be called a client and the program for the service provision will be called a server. However an individual server may act as a client to another server so they will not be clients and servers in the currently accepted meaning of these words each is a component in a distributed application. Each component can be designed to act as both a client and a server and each abstract client or server can hold within it a hierarchy of clients and servers.This concept is similar to a single program that calls procedures or functions which in turn call other procedures or functions. It differs in that each component is a separately compiled unit with a known interface. The glue that connects them one with another is a procedure call that is not limited to the functions within a component but can use procedures that reside elsewhere on the same machine or even on a different machine.

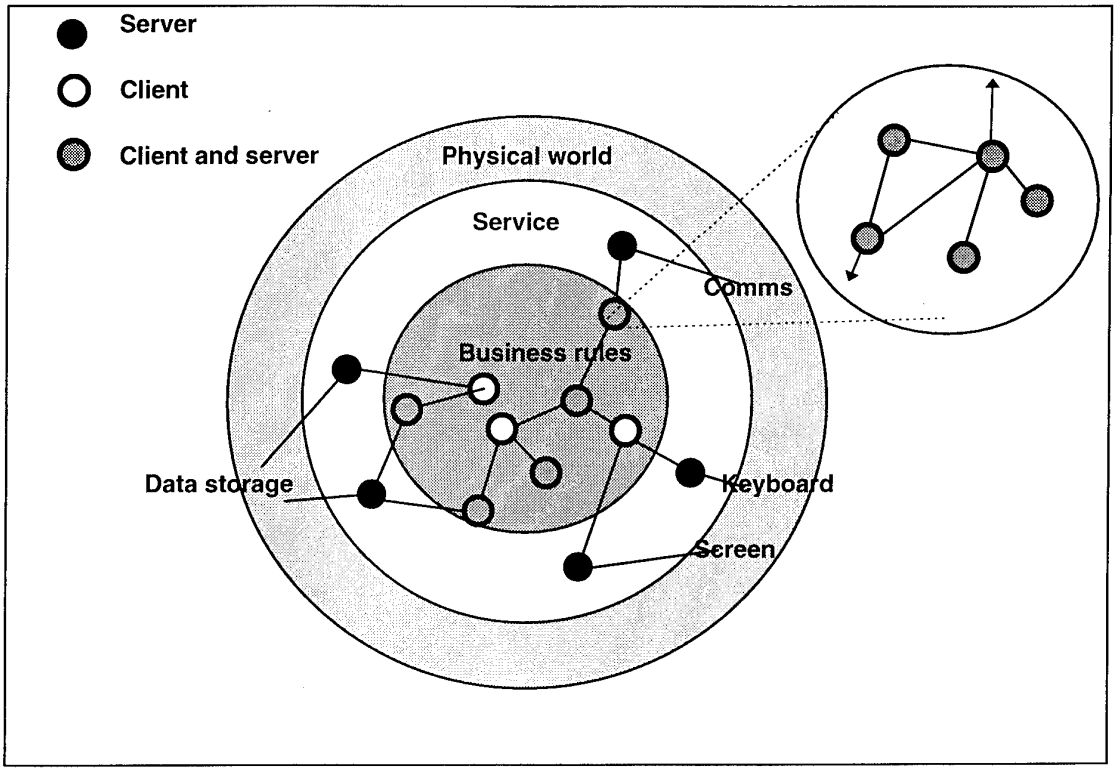


Figure 6. A three ring architecture.

If a change is made within a component then that component needs to be recompiled but no other because the interface is maintained. So if a bug is fixed, a new faster algorithm implemented or even a different vendor's DBMS is introduced only that component need be recompiled.

This architecture also permits customisation of a system by including particular versions of a facility; for example the interface a wordprocessor component uses to call a spelling checker component remains the same whether it calls an English or a Spanish version. Applications, such as wordprocessors, can be constructed from a small, kernel client component and many subordinate components. If a new facility is required, a new component is created to provide the facility and the client component is modified to call the new component; in this case both the modified client component and the new server component need to be recompiled but any subordinate components called by either the client or the "server" component need not be changed. The communication through a single interface enforces the good software engineering practices of separation and encapsulation and minimises the problems of errors occurring when software is changed.

The concentric architecture also provides the potential for the construction of high availability systems. Where server components are replicated, clients can be constructed to respond to communications or server failure by seeking out a replica server and connecting to the replica. Experiments have shown that it is possible to continue processing in the event of a failure with only a short delay and without intervention by user or system administrator.

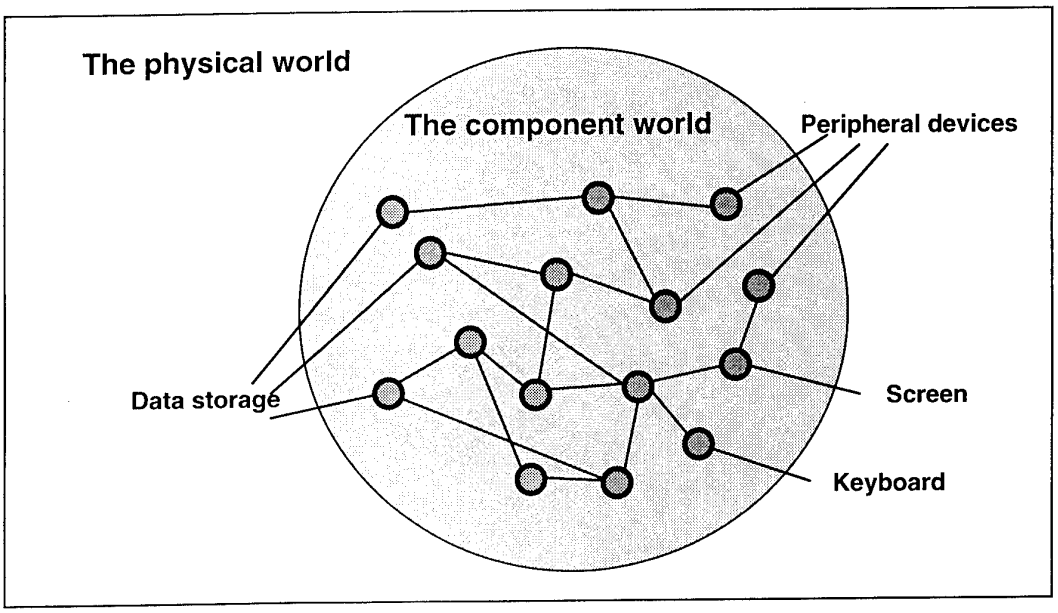


Figure 7 A single ring architecture.

The single ring architecture diagram (Figure 7) depicts a static view of the dynamic state of the system; over a period of time a client will connect to a server then disconnect, then another client will connect to the same server. So any view will be a snapshot of a constantly changing system. The clients and servers need not be on the same host machine, indeed they may be separated by thousands of kilometres and be running on different vendor's machines. The dynamic view also depicts the situation where many clients may be attached to a single server and where there are several invocations of a single server or a single client. When introducing a new component into the system a mechanism called a trader or broker may be consulted to discover where the resources needed by the component are located; the trader keeps track of what procedures are on offer and where they may be obtained.

Thus we have lots of little "machines" that have input and output and process data. They can be replicated and dynamically combined in various ways to suit the needs of the user. In a sense they are all servers and when combined each do their little bit for the structure. The only specialist machines are those that communicate with the physical world. The structure's closest analogy is probably an organism, in that it has the ability to evolve; growing, changing and, partly, dying as needs change.

The essence of minimising the ramifications of change is that the granularity of the component is very small. The overhead will be high - but the benefits will also be high.

4.2 Implementation

In order to create a practical implementation of such a system a standard is required that provides a number of basic facilities. The most important is an open remote procedure call facility that operates across any vendor's host machine and operating system. Another is an interface definition language (IDL) which will permit developers of components to specify the interface to a specific component so that others can use that information in designing calls to that component. An IDL complier is also required so that the interface definition can be machine readable. The third essential requirement is the ability of one "server" to handle several "client" calls simultaneously; this needs a mechanism within the server to keep each client's activities separate from the others. Candidates for this standard are the Open Software Foundation's Distributed Computing Environment (DCE) and the Object Management Group's Common Object Request Broker Architecture (CORBA). At the time of writing DCE has been implemented by all the major hardware vendors and a few independent software vendors whereas the availability of CORBA is more limited. Moreover, as CORBA does not define an implementation standard, interoperability is not guaranteed. DCE is therefore currently the most obvious choice for an implementation standard.

4.3 Pros and Cons of the Concentric Architecture

The architecture described has some disadvantages but provides many advantages in a changing world.

Disadvantages:

- Separation of the software into independent components incurs an overhead every time the component is invoked.
- Communication bandwidth limits the speed of the system.

Advantages:

- Proven systems need not be discarded as requirements change, saving time and money.
- High availability systems can be constructed.
- The location of computing resources are transparent to the user.
- Systems are scalable, as requirements change more or less components are provided. These components can run on any machine, large or small, or on multiple machines. When DCE is used the components can run on any of the major vendor's operating systems.
- Systems may be physically spread over a wide geographic area but are logically local.
- When DCE is used privacy and security can be implemented network wide.
- Applications can be customised.
- Error correction and code modification can be carried out without disruption.

It would seem that the advantages of the single ring architecture outweigh the disadvantages, particularly when it is viewed against the proprietary two and three layer architectures.

5. Conclusions

Over the past twenty years information systems architectures have migrated away from the centrally located mainframe in favour of geographically distributed services. The disadvantages of a distributed system are poorer reliability, interoperability and security. Distributed Computing Environment (DCE) is a software standard and technology which provides a very high level of reliability, interoperability and security within distributed information systems.

DCE has been in development for about five years but has only been widely used in the last two years. It consists of a number of services that have been selected from current proprietary facilities and integrated so that they work together. DCE is widely recognised as a tool for providing interoperability between heterogenous computer systems. It also has the potential to provide computer systems which can more readily adapt to change.

There are three basic elements of a computer program: the user interface, the business logic and the data . As programs have increased in size and complexity, tools and standards have emerged to support these three elements as separate entities. Mapping the logical view of a computer program to a client/server architecture, such as DCE, requires a decision to be made on which elements are clients and which elements are servers.

A three layer client/server architecture can have the business rules vested in the client and with servers supporting the interface and function (usually database) services. The clients are created by people who are familiar with the organisational or business procedures whilst the expertise used to create the servers is that of technical experts. By breaking the system down into parts the ramifications of change are minimised as each part may be changed individually.

The three layer client/server architecture can be viewed as three concentric rings. A concentric view highlights the potential of the three layer architecture to support change. In the outer ring lies the physical world, in the middle ring the service providers and the centre the business rules. The concentric architecture can be viewed as lots of little "machines" that have input and output and process data. They can be replicated and dynamically combined in various ways to suit the needs of the user. This view leads to a final view of the architecture as a single ring.

The essence of minimising the ramifications of change is that the granularity of the component is very small. Although this leads to high overheads they may be far outweighed by the benefits.

Acknowledgments

The work reported in this paper has been supported in part by the Distributed Systems Technology Cooperative Research Centre.

Appendix A

Overview of the OSF Distributed Computing Environment(DCE)

The Open Software Foundation is a not for profit organisation that uses consensus to create a standards for various vendor independent concepts, Motif and DCE are two of these standards. The DCE standard provides facilities to build open, distributed computer systems. The key parts of the standard are:

- **Remote procedure calls.**

Remote procedure calls permit the application programmer to use functions from separately compiled programs, that may be running on a different platform many kilometres away. These procedures are called in the same manner as procedures local to the program and DCE handles all the communication problems transparently to the programmer. The local program is called the client and the remote program is called a server. A server may access a database or carry out a complex parallel computation but all the application programmer needs to know is the name of the procedure and its parameters (see Figure A1).

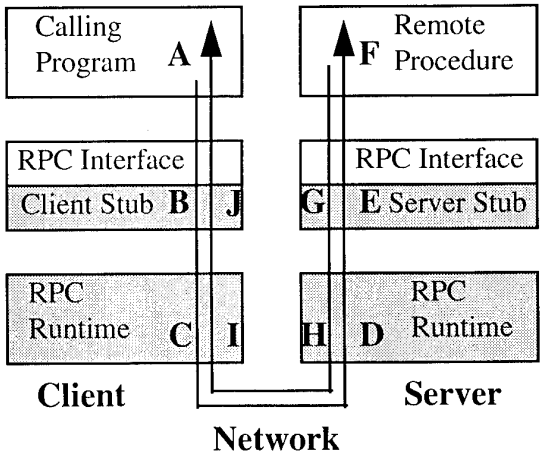


Figure A1. The path taken by a remote procedure call from the calling program to the remote procedure and back again.

- **An interface definition language.**

The interface definition language allows the interface to each server to be defined in a standard way so that any client may use its services. A programmer writes the interface definition file using the interface definition language. An interface definition language compiler produces header files that support the data types in the interface definition and generates the client and server stub files.

A universal unique identifier (UUID) is used to distinguish the interface. A UUID is produced using a UUID generator utility (see Figure A2).

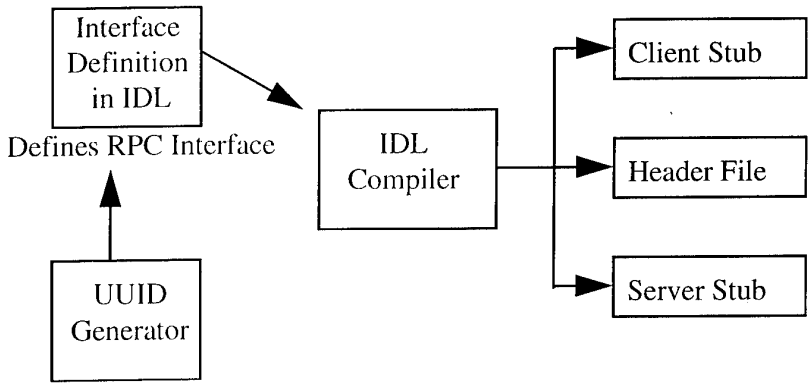


Figure A2. Defining the interface to each server requires a unique identifier to be generated.

- **POSIX standard threads.**

Servers may need to process the requirements of several clients at one time without the needs of one being entangled with another. As the server is a single program it needs a mechanism to run several processing streams simultaneously; this mechanism is called threads (see Figure A3). The version of threads adopted by DCE is that from the POSIX standard.

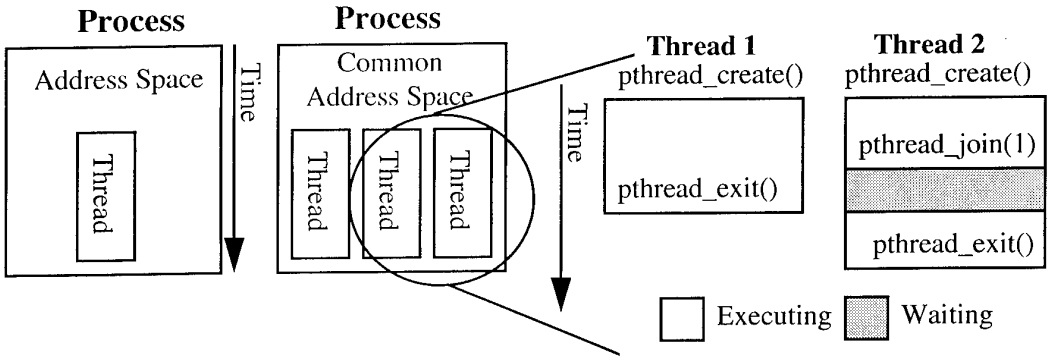


Figure A3. Threads permit servers to respond to multiple clients.

- **A local cell directory service.**

DCE connects together a logical group of machines into a cell. The machines in a cell may be few or thousands, they can all be local or they can be spread around the world. However, all machines in a cell share the same file system so all the users of the machines in the cell can see the same files. Of course, if the file the user wants is on a machine on the other side of the world it will take longer to access it than if it were local but the user need never know where it physically resides. This access is governed only by the user's authorised access rights. (see para. on security below)

- **Access to the global computing environment via X.500 or the Domain Name Service.**

Whilst most of a user's work will be done within the cell it is natural that occasionally there will be the need to access machines outside the cell. A user can reach any machine outside the cell via the X.500 directory standard or via the Domain Name Service. If the required machine is in another DCE cell the user can become a principal within that cell with whatever access rights granted by the cell administrator. A Global Directory Agent (GDA) makes cell interaction possible. When a cell directory service determines that a name is not in its own cell, it passes the name to the GDA. The GDA searches the appropriate global naming environment for more information about the name.

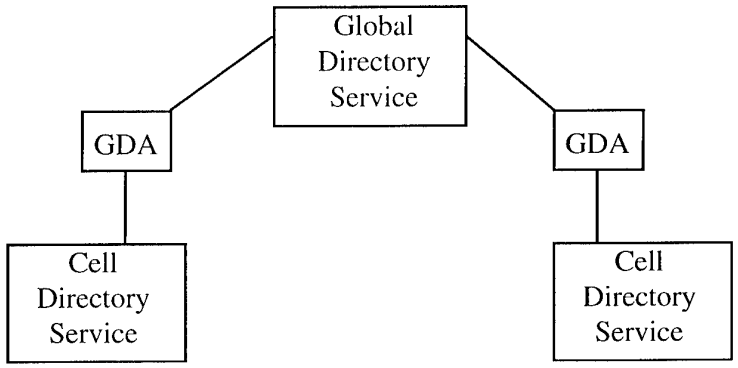


Figure A4. Access to machines outside of a cell is gained through a global directory agent (GDA).

- **A distributed time service.**

DCE provides a distributed time service to synchronise the time on the machines within a cell. Without this synchronism there could be chaos as locally each machine may have a different system time and some machines may even be in different time zones. For the purposes of security, file updating, transaction processing, backing up, compilation, etc. it is crucial that all machines in a cell have the same reference time.

- **A security service.**

DCE provides a number of facilities to provide security on machines with no security such as MS Windows and OS/2 machines and enhance security on Unix machines. It uses an enhanced form of Kerberos to authenticate a user on login and when they access restricted programs. It also gives six levels of access control to partition data and databases.

- **System wide administration tools.**

Administration of distributed systems, particularly those spread over a wide area, is complex and difficult. DCE provides a set of cell wide administration tools to ease the task.

Appendix B

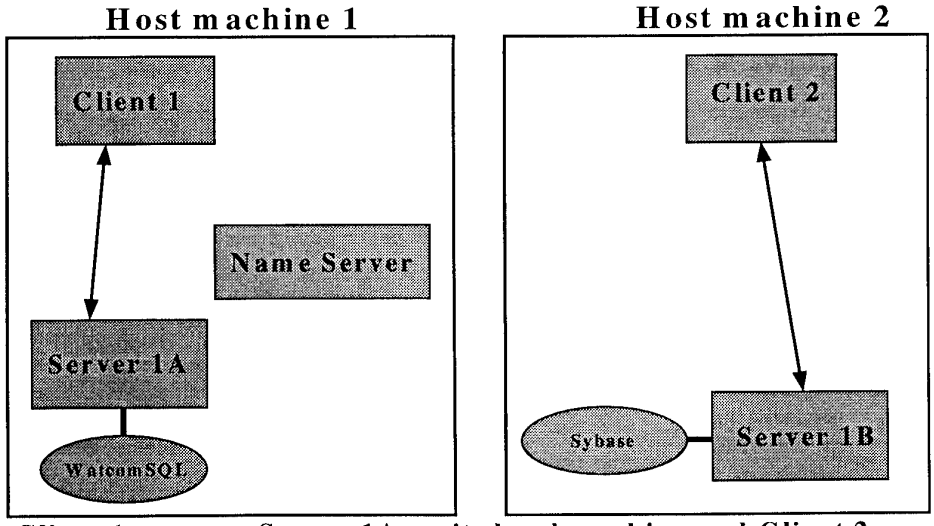
DSTO experience with DCE

DSTO has conducted an experiment that demonstrates it is possible to create a cell composed of six heterogenous machines with some in Salisbury, South Australia and some in Canberra. The machines were a H-P E55, two Digital Alpha 3000, a Sun Sparcstation, two OS/2 PCs and a Windows 3.11 PC(client only). DCE clients running on one machine were able to access DCE servers running on any of the other machines. Thus demonstrating interoperability.

As part of this experiment the client and server programs were ported from machine to machine quickly and without major problems. The exception to the pattern was the OS/2 machine where a little more effort was required as the signal mechanisms were different to that for Unix. This demonstrated the portability of DCE code across proprietary platforms.

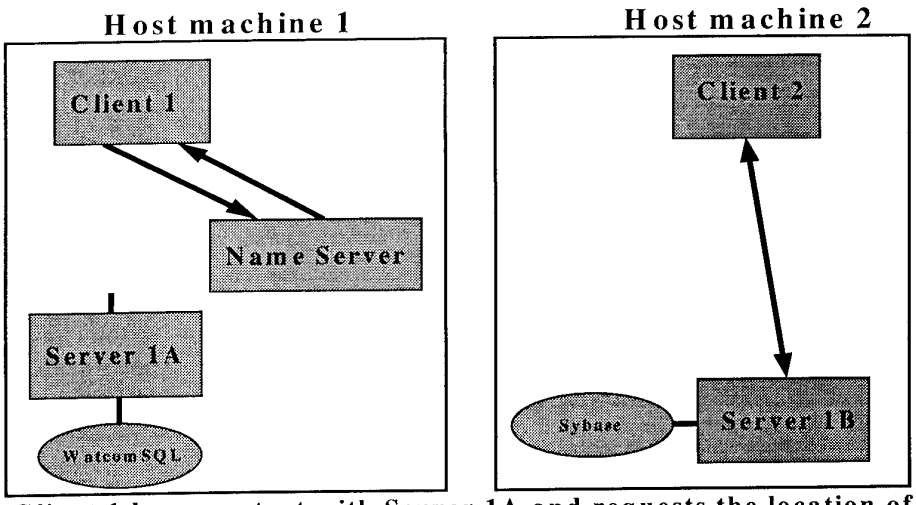
A second experiment was conducted to test the potential survivability of DCE systems. A DCE database server was replicated on several machines. A client was written to access the server. In addition to the application code, whenever a call to the current server was made the client checked if the call had succeeded or failed. If the call failed the client asked the local name server for the location of a replica of the server, bound itself to the new server and continued processing.

In the experiment several instances of the client were set running and initially each one accessed a server local to it. A server process was killed and it was observed that the client formerly attached to that server recommenced processing in a matter of seconds.



Client 1 accesses Server 1A on its local machine and Client 2 accesses Server 1B, a replica of Server 1A, on its local machine.

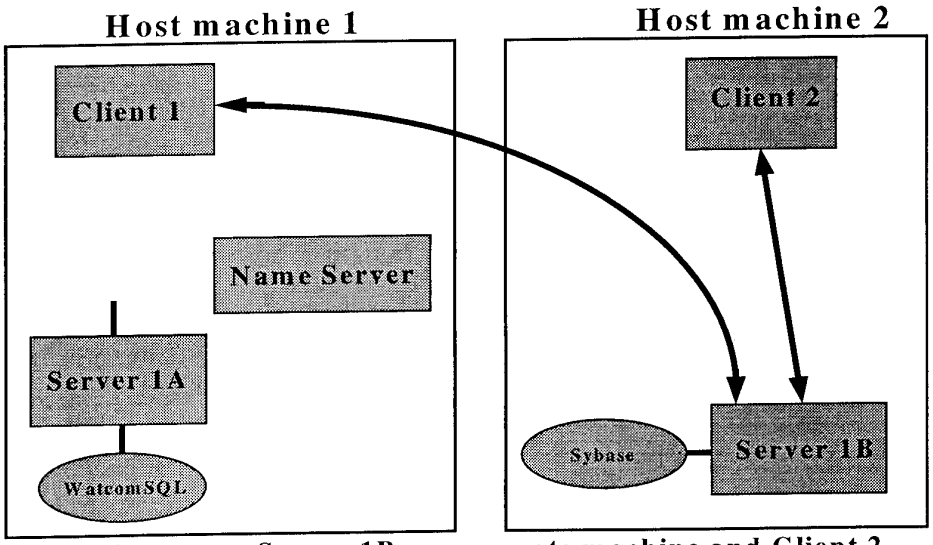
Figure B1.



Client 1 loses contact with Server 1A and requests the location of a replica from the Name Server. It is given the location of Server 1B.

Figure B2.

Where the only available server was geographically distant a poorer response was noted but the client application was still able to continue processing without assistance from a systems administrator



Client 1 accesses Server 1B on a remote machine and Client 2 continues to accesses Server 1B on its local machine.

Figure B3.

This demonstrated that it is possible to write DCE based applications that are fault tolerant, in that they can lose connection with their primary server and still continue processing. It should be noted that whilst the database servers were replicated and the information was replicated the DBMSs were different.

Distributed Computing Environment: An Architecture for Supporting Change?

J. Mansfield and J. Clothier

(DSTO-CR-0007)

DISTRIBUTION LIST

	Number of Copies
Defence Science and Technology Organisation	
Chief Defence Scientist and members of the)	1 shared copy
DSTO Central Office Executive)	for circulation
Counsellor, Defence Science, London	(Document Control sheet)
Counsellor, Defence Science, Washington	(Document Control sheet)
Senior Defence Scientific Adviser)	1 shared copy
Scientific Adviser - POLCOM)	
Director, Aeronautical & Maritime Research Laboratory	1
Electronics and Surveillance Research Laboratory	
Chief Information Technology Division	1
Research Leader Command & Control and Intelligence Systems	1
Research Leader Command, Control and Communications	1
Research Leader Military Computing Systems	1
Head Command Support Systems Group	1
Head Information Management Group	1
Manager Human Computer Interaction Laboratory	(Document Control sheet)
Executive Officer, Information Technology Division	(Document Control sheet)
Head Software Engineering Group	(Document Control sheet)
Head, Trusted Computer Systems Group	(Document Control sheet)
Head, Advanced Computer Capabilities Group	(Document Control sheet)
Head, Command Support Systems Group	(Document Control sheet)
Head, Intelligence Systems Group	(Document Control sheet)
Head, Systems Simulation and Assessment Group	(Document Control sheet)
Head, Exercise Analysis Group	(Document Control sheet)
Head, C3I Systems Engineering Group	(Document Control sheet)
Head, Computer Systems Architecture Group	(Document Control sheet)
John Mansfield, (Author) CSSG, ITD	1
Dr Jennie Clothier	1
Publications and Publicity Officer, ITD	1

Strategy and Intelligence	
Assistant Secretary Scientific Analysis	1
Principal Research Scientist (R&D)	1
Assistant Secretary, Information Technology	1
HQADF	
Director General, Force Development (JOINT)	1
Director General, Force Development (SEA)	1
Director General, Force Development (LAND)	1
Director General, Force Development (AIR)	1
Director, Operational Information Systems	1
DD-EW	1
Navy	
Navy Scientific Adviser (NSA)	1
Army	
Scientific Adviser, Army (SA-A)	1
Project Director, AUSTACSS	1
Air Force	
Air Force Scientific Adviser (AFSA)	1
CO, Electronic Warfare SQN	1
ACQUISITION AND LOGISTICS PROGRAM	
Director General, Information Management	1
Project Director, ADFDIS	1
Project Director, JP2030	1
Project Director, AUSTACCS	1
Libraries and Information Services	
Defence Central Library - Technical Reports Centre	1
Manager Document Exchange Centre (MDEC) (for retention)	1
Additional copies which are to be sent through MDEC	
DIS for distribution:	
National Technical Information Centre. United States	2
Defence Research Information Centre, United Kingdom	2
Director Scientific Information Services, Canada	1
Ministry of Defence, New Zealand	1
National Library of Australia	1
Defence Science and Technology Organisation Salisbury, Research Library	2
Library Defence Signals Directorate Canberra	1
AGPS	1
British Library Document Supply Centre	1
Parliamentary Library of South Australia	1
The State Library of South Australia	1
Spares	
Defence Science and Technology Organisation Salisbury, Research Library	6

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION
DOCUMENT CONTROL DATA

1. PRIVACY MARKING/CAVEAT (OF DOCUMENT)
N/A

2. TITLE

Distributed Computing Environment:
An Architecture for Supporting Change?

3. SECURITY CLASSIFICATION (FOR UNCLASSIFIED REPORTS THAT ARE LIMITED RELEASE USE (L) NEXT TO DOCUMENT CLASSIFICATION)
Document (U)
Title (U)
Abstract (U)

4. AUTHOR(S)

J. Mansfield and J. Clothier

5. CORPORATE AUTHOR
Electronics and Surveillance Research Laboratory
PO Box 1500
Salisbury SA 5108

6a. DSTO NUMBER	6b. AR NUMBER	6c. TYPE OF REPORT	7. DOCUMENT DATE
DSTO-CR-0007	AR-009-456	Client Report	November 1995

8. FILE NUMBER	9. TASK NUMBER	10. TASK SPONSOR	11. NO. OF PAGES	12. NO. OF REFERENCES
N9505/10/4	ADL 94/150	DGFD (Joint)	34	N/A

13. DOWNGRADING/DELIMITING INSTRUCTIONS
N/A

14. RELEASE AUTHORITY
Chief, Information Technology Division

15. SECONDARY RELEASE STATEMENT OF THIS DOCUMENT

Public Release

OVERSEAS ENQUIRIES OUTSIDE STATED LIMITATIONS SHOULD BE REFERRED THROUGH DOCUMENT EXCHANGE CENTRE, DIS NETWORK OFFICE, DEPT OF DEFENCE, CAMPBELL PARK OFFICES, CANBERRA ACT 2600

16. DELIBERATE ANNOUNCEMENT

No limitation

17. CASUAL ANNOUNCEMENT

Yes

18. DEFTEST DESCRIPTORS
Distributed Computing Environment
Computer systems
Computer programs
Client/server computing

19. ABSTRACT

Distributed Computing Environment (DCE) has been in development for about five years but has only been widely used in the last two years. It consists of a number of services that have been selected from current proprietary distributed facilities and integrated so that they work together. Together these services form an architecture for distributed computing that enables users to carry out the new, cheaper operations they require with the interoperability, reliability and security standards of mainframe computers.

The facilities provided by DCE are often reviewed from either the perspective of reliability or the perspective of interoperability. This paper reviews the facilities of DCE from the perspective of change. An architecture is proposed which will support the evolution of information systems.