Dionisis X. Adamopoulos, George Pavlou, Constantine A. Papandreou and Emmanuel Manolessos

Distributed Object Platforms in Telecommunications: A Comparison Between DCOM and CORBA

Continuous advances in telecommunications technology coupled with the development of powerful desktop workstations, and increased user demands for complex service provision chains, integrated service offerings, and new sophisticated telecommunications services with multimedia characteristics are fueling the growth of object-oriented distributed computing. Recognising the growing importance of distributed object platforms in telecommunications service engineering, this paper attempts to compare the two most important of them; namely Microsoft's COM/DCOM and OMG's CORBA. After a brief overview of both architectures, a decision framework is proposed by identifying a set of core and service engineering related properties and examining the way that DCOM and CORBA supports these properties. Based on the proposed decision framework, some conclusions are drawn about the suitability of each platform for different service development requirements.

Introduction

There are many driving forces which have compelled telecommunications operators and vendors to seek new solutions in telecommunications service engineering. Among them, growing competition and the progressive convergence of information and telecommunications technologies has led to an increased focus on how a great variety of advanced multimedia telecommunications services (telematic services) with enhanced functionality can be efficiently and effectively developed and deployed in shorter time-frames taking advantage of different network technologies, end-systems, communications protocols, operating systems, and programming language environments. Distributed object platforms have been recognised as a key technology solution to this problem, mainly due to recent developments in object orientation and distributed computing.

Upon these platforms, which are actually object-oriented distributed processing environments (DPEs), telecommunications services are isolated by the underlying computing and networking technology and are realised by a (possibly large) set of interacting service objects/components, which are distributed across different network elements. Currently the two most important available distributed object platforms are Microsoft's (Distributed) Component Object Model (COM/DCOM) and the Common Object Request Broker Architecture (CORBA), which is supported by the Object Management Group (OMG).

This paper examines DCOM and CORBA as the current key general-purpose distributed object-oriented environments. A comparison between them is attempted based on a set of carefully selected core and service engineering related properties, and as a result a decision framework is proposed with the objective to guide service designers/developers during the selection process.

Distributed Objects in Telecommunications

As the telecommunications environment is gradually changing its face towards an open market of information services, it is becoming apparent that major private and public networks are actually large distributed object systems. These systems are populated by a dispersed set of objects that can request services from one another through a communications mechanism, using interfaces defined in a consistent interface definition language (IDL). The most important benefits offered by distrib-

Dionisis X. Adamopoulos:
Centre for Communication Systems Research
School of Elect. Eng. and Mathematics
University of Surrey, England
D.Adamopoulos@ee.surrey.ac.uk
Prof. George Pavlou:
Centre for Communication Systems Research
School of Elect. Eng. and Mathematics
University of Surrey, England
D.Adamopoulos@ee.surrey.ac.uk
Dr. Constantine A. Papandreou:
Hellenic Telecommunications Organisation (OTE)
Athens,
Greece
kospap@org.ote.gr
Emmanuel Manolessos:
IBM Hellas
274 Kifissias Avenue
Athens,
Greece
manoleso@gr.ibm.com

uted object technology to telecommunications systems are ease of development and maintenance, abstraction, modularity, reusability, and granularity flexibility [8, 9, 14].

Distributed object technology is already in use in the telecommunications world[2, 14]. A characteristic example is the telecommunications management network (TMN) development (especially its information model), and an example where this influence is maximised is the telecommunications information networking architecture (TINA), standardised by the TINA Consortium (TINA-C). The main objective of TINA-C is to define and validate an innovative architectural framework (a long-term service architecture) that will address in an integrated manner service control and service management. TINA-C services are considered as a set of distributed computational objects that operate on a distributed object platform[13].

Microsoft's COM/DCOM

COM constitutes the foundation of Microsoft object services and has been assigned to the Open Group for standardisation[10]. Distributed COM (DCOM) is the distributed extension to COM that builds an object remote procedure call (ORPC) layer on top of DCE RPC to support remote objects. DCOM makes COM objects location-independent, and adds security and multithreading to COM[3].

A DCOM server can create object instances of multiple object classes. Each DCOM server object can support multiple interfaces, each representing a different view or behaviour of the object. An interface consists of a set of functionally related methods. A DCOM client interacts with a DCOM object by acquiring a pointer to one of the server object's interfaces. Thus, it invokes the server object's exposed methods through the acquired interface pointer as if the server object resided in the client's address space.

As long as a platform supports COM services, DCOM can be used on that platform. DCOM is extensively supported on the Windows platform. Companies like Software AG provide DCOM service implementations through their EntireX product for Unix, Linux and mainframe platforms, Digital for the Open VMS platform, and Microsoft for the Solaris platform.

OMG's CORBA

CORBA is supported by the Object Management Group (OMG) as part of an initiative to develop a comprehensive object management architecture (OMA) for object-oriented computing[11]. CORBA adopts an object-oriented approach. Object interfaces are described in terms of an implementation language-neutral IDL. CORBA has a special component, called the *Object Request Broker* (ORB), which is responsible for making object distribution transparent and providing a mechanism for trading, enabling object requests to be carried out in a heterogeneous distributed environment.

Besides CORBA, OMA also defines certain key object interfaces. These can be divided into the lower-level CORBA services and the higher-level CORBA facilities.

Table 1 Comparing DCOM and CORBA: Basic characteristics

Basic Characteristics	DCOM	CORBA
Inheritance of a base interface	Every object implements *IUnknown*	Every interface inherits from *CORBA.object*
Unique identification of a remote server object	Through its interface pointer	Through an object reference (objref)
Unique identification of an interface	Using the concept of interface IDs (IIDs)	Using the interface name
Unique identification of a named implementation of a server object	Using the concept of Class IDs (CLSIDs) the mapping of which is found in the registry	By the mapping to a name in the implementation repository
Reference generation of the remote server object	Performed on the wire protocol by the Object Exporter	Performed on the wire protocol by the Object Adapter
Handling of common tasks like object registration, skeleton instantiation, etc.	Either explicitly performed by the server program or handled dynamically by the DCOM run-time system	Performed implicitly by the constructor
Underlying remoting protocol	Object Remote Procedure Call (ORPC)	Internet Inter-ORB Protocol (IIOP)
Activation of a server object	Mainly by using *CoCreateInstance()*	Mainly by binding to a naming or a trader service
Mapping of object name to its implementation	Handled by the registry	Handled by the implementation repository
Storage of type information	Type library	Interface repository
Client side stub	Called a proxy	Called a proxy or stub
Server side stub	Called a stub	Called a skeleton
Definition of parameters passed between the client and server objects	Defined in the interface at the interface definition file. Depending on what the IDL specifies, parameters are passed either by value or by reference	All interface types are passed by reference. All other objects are passed by value including highly complex data types
Definition of complex types	Complex types that will cross interface boundaries must be declared in the IDL	Complex types that will cross interface boundaries must be declared in the IDL
Support of distributed garbage collection	On the wire by a pinging mechanism which garbage collects remote object references and encapsulates them in the *IOXIDResolver* interface	No
Platform support	Any platform as long as there is a COM service implementation for that platform	Any platform as long as there is a CORBA ORB implementation for that platform
Programming language support	Since the specification is at the binary level, diverse programming languages can be used	Since it is just a specification, diverse programming languages can be used, as long as there are ORB libraries suitable for coding in a specific language

CORBA services define such object interfaces as naming, life cycle, persistence, transaction, concurrency control, relationship, time, and security. CORBA facilities provide horizontal and vertical application frameworks, by defining collections of facilities that processes may use through CORBA objects, such as compound documents, user interfaces, and system management[12].

Comparing DCOM and CORBA

Taking into account the basic characteristics of DCOM and CORBA, as were presented in the previous sections, Table 1 represents an initial attempt at comparing the two technologies. This table reveals the wide scope and the richness of both platforms, and it is believed to be much more concise and informative than other comparison attempts found in the literature, which are based extensively on code examples[4, 6]. These code examples, while reasonable and correct, are extremely limited and involve choices among a variety of possible approaches. Furthermore, it is very difficult to keep the two implementations exactly equivalent. Therefore, comparisons based heavily on code examples can be used only as a means to become familiar with DCOM and CORBA, and not as a basis for general conclusions about either technology.

In order to derive such desired conclusions, a set of core properties that should characterise every distributed object platform are identified. Table 2 summarises the way that DCOM and CORBA supports each of these properties, and offers an insight on the capabilities of the platforms pertaining to their use in practical situations. It is explained in more detail in the subsection on 'Core Properties'.

Furthermore, a set of service engineering related properties is also identified, and their support by DCOM and CORBA is summarised in Table 3. This table focuses on how DCOM and CORBA provide a solution for developing effective (possibly large-scale) telematic services, and how they assist in the deployment of these telematic services across the Internet, within an intranet, over an extranet, or simply with a web front end.

During this comparison attempt, the various value-added services provided by DCOM and CORBA are considered. These include, for DCOM, the Microsoft Transaction Server (MTS), the MicroSoft Message Queue server (MSMQ), the Microsoft Cluster Server (MCS), and the Microsoft Management Control (MMC), and for CORBA (2.0), the naming, events, life cycle, persistence, relationship, externalisation, transaction, concurrency, property, licensing, time, trader, and security services (CORBA services)[7, 12]. Table 3 is explained in more detail in the subsection on 'Service Engineering Related Properties'.

Core properties

Object locator
A mechanism by which objects can be located and subsequently activated is necessary. DCOM allows for a locally maintained object locator on the server machine using object names, while CORBA and MTS centralise the locator on a single (or perhaps a few) domain machines that can identify object servers in the domain. More specifically, the most well-known object locator in DCOM is the registry. It maps a CLSID (or a readable name called *ProgID*) to the path name of the server executable that supports the CLSID. However, the registry is consulted only after the Service Control Manager (SCM) has failed to locate any running object instance. In CORBA, an object can locate another object in a system, by using either the naming service or the trader service.

Server activation
A DCOM object server is not necessary to be running when a client request is made to instantiate an object. DCOM locates the server code through the registry, and will start the server using SCM. DCOM also

Table 2 Comparing DCOM and CORBA: Core Properties

Basic Characteristics	COM / DCOM	CORBA
Object Locator	Locally maintained	Centralised in the domain
Server Activation	Yes (Service Control Manager)	Yes (Basic Object Adapter)
Data Typing	Strong and predetermined (vtable method)	Strong and predetermined (Static Interface Invocation, SII)
Dynamic Invocation	Dispatch interface	Dynamic Interface Invocation (DII)
Communication Type	Synchronous, Asynchronous (callback support)	Synchronous, Asynchronous, Deferred Synchronous
Inheritance	Interface, Implementation (containment, aggregation)	Implementation
State Persistence	Yes (2 models)	Yes (Persistence Service)
Load Balancing	No	No
Exceptions Handling	Not directly (error reporting)	Yes (CORBA IDL)
Multithreading	Yes (2 models)	Yes

Table 3 Comparing DCOM and CORBA: Service Engineering Related Properties

Service Engineering Related Properties	COM / DCOM	CORBA
Scalability	MTS, Active Directory Service Interface (Win NT 5.0)	Naming service, Trader service
Reliability	MTS, MCS, MSMQ	Transaction service
Security	Built-in: NT LAN Manager, MTS, MS Crypto API, Authenticode SDK	Platform dependent: 3 security levels (0, 1, 2)
Manageability	MMC	Vendor specific tools, Transaction service
Support for Web-based Telematic Services	ActiveX, MS Active Server Page Technology	JavaScript/Java
Support for Internet/Extranets	Two-factor authentication, Remote Data Service (RDS)	Two-factor authentication, Secure Socket Layer (SSL)
Support for Intranets	Desktop tools, ActiveX, Active Data Objects, MSMQ	Desktop tools (via a bridge), Event service, Persistence service

allows access to servers that are already running when the client request is made, as running objects are registered with the Running Object Table (ROT). In CORBA, server activation is handled by the Basic Object Adapter (BOA). If a client makes a request for an object that is not running, then the BOA finds the server and launches it to create the object. In this process it uses the implementation repository, which holds information about the location of every server object.

Data typing
Once an object has been located and activated, the client will need to be able to communicate with it. Strong data typing is supported by both DCOM and CORBA through the use of interfaces. In the static method of invoking operations on DCOM objects, the MIDL compiler, based on the IDL definition of the object and its interfaces, creates the corresponding proxy and stub code. Due to the way that the static invocation is implemented, this is often referred to as the *vtable* method for invoking objects. In CORBA, in the case of the static interface invocation (SII), all methods are specified in advance and are known to the client and server through the stubs and skeletons that are produced by the IDL compiler.

Dynamic invocation
Although strong and predetermined data typing is extremely important for the creation of complex and robust code, sometimes the flexibility of slightly looser typing, similar to the kind which is important to interpreted scripting languages, is necessary. This can be allowed by providing dynamic querying of objects for the functionality that they support. DCOM provides this facility through its IDispatch interface, and CORBA through its DII mechanism. In essence, predetermined typed interfaces are used that allow a dynamic interface to be queried[15].

Communication type
The communication between objects can be either synchronous or asynchronous. DCOM is mainly synchronous. However, it allows for flexible callback mechanisms, such as connection points, to be implemented. In CORBA, a client can invoke a method, either synchronously or post it asynchronously. Posting means that the calling object is not blocked waiting for the reply. Instead, it can specify which of its methods the response should invoke. It has to be noted that the receiver cannot tell the difference between a synchronous or an asynchronous call.

Inheritance
DCOM allows interface inheritance, whereas CORBA allows implementation inheritance. In interface inheritance when one interface is derived from another, the derived interface must supply an implementation for the methods of the base interface; all it inherits is the responsibility to supply the interface. In implementation inheritance a derived interface inherits the interface and an implementation. DCOM provides a similar mechanism using containment or aggregation[7].

State persistence
Objects represent both functionality and data. A client wishing to access an object would typically create the object, access its services, and then destroy it. The object server needs to be able to associate a client connection with a particular object, since each client has some assumption about the state of the object when it last accessed it.

Both DCOM and CORBA use the notion of saving object state for later reactivation. DCOM has two persistence models. The original model requires that objects implement an interface that supports persistence using one of several known storage media (file, stream or storage). A more recent persistence model in MTS provides server-managed storage. In CORBA, the persistence mechanism is completely transparent (persistence service). The client has no legitimate means of determining where or how an object is stored (unless some object with knowledge of the storage details provides an interface with a method that divulges the information). The implementation is exclusively responsible for managing persistence[12].

Load balancing
A server machine may provide several object servers, and each of these may provide several object types. Thus, the server machine may become a bottleneck in the distribution of objects, and this leads to the need for load balancing. This facility is not offered currently, neither in DCOM, nor in CORBA. Generally, load balancing is an area that has little available support in the mainstream distribution framework at present, but intensive development is currently underway[15].

Exceptions handling
DCOM has a standard way of handling error data through the return of a 32 bit error code, called an *HRESULT*, by all methods. At the language/tool level, a set of conventions and system provided services (the IErrorInfo object) allows failure HRESULTs to be converted into exceptions in a way natural to the programming language. On the other hand, CORBA specifies an extensible exception capability that maps naturally into languages that have native exceptions, like C++ and Java, and that maps into exception data in languages that do not. It is based on user-defined exception types declared in CORBA IDL.

Multithreading
DCOM supports multithreaded server objects, but it requires that the DCOM libraries be initialised in the threads that use them. There are two main models (the *apartment* model and the *free threading* model). A third model (still to be released) is called the *rental* model. In this model, which will be used by the MTS, one thread 'helps' another, in a fashion that still behaves as if the object is single threaded. CORBA object servers can also be multithreaded. Issues such as (for example) whether an object is created in a new process or in a new thread are handled by the ORB through the object adapter.

Service engineering related properties

Scalability
MTS provides a set of DCOM interfaces and libraries that allow telematic services to easily scale as the number of users and user data increase. With the forthcoming Windows NT 5.0, DCOM will obtain the Active Directory Service Interface (ADSI), which will allow components to seamlessly use a variety of existing naming services, such as NetWare Directory Service (NDS), Lightweight Directory Access Protocol (LDAP) or even the Windows registry. In that way, a telematic service will be able to handle an increasing number of geographically dispersed users.

In CORBA, the Object Activation Daemon (OAD) and the implementa-

tion repository allow efficient use of resources by only instantiating objects when required. The centralised naming service provides location independence for applications and their users, while the trader service allows more sophisticated component searches. Static load-balancing among replicas of an application is available via the naming service.

Reliability
Distributed objects should offer transparency to a client, and part of this transparency is the guarantee that the object connection will be reliable throughout the client's use of an object. In the optimum case, a remote object must be as reliable as a local object.

Reliability can be achieved by using a transaction monitor, like MTS or the OMG transaction service. MTS allows telematic services to use distributed transactions to reliably update data across disparate data stores, while the OMG transaction service supports an Open Group Distributed Transaction Processing (DTP)-compliant model for distributed transactions. As far as DCOM is concerned MCS and MSMQ also increase reliability.

Security
DCOM has been designed with security built in, while CORBA objects can implement their own security mechanisms for the platform on which they are implemented. More specifically in DCOM, the NT LAN Manager (NTLM) and the MTS authenticate users and authorise checking via Access Control Lists (ACLs). Additionally, the MS Crypto API provides data encryption and integrity to prevent eavesdropping and tampering, while the Authenticode SDK uses digital signatures to provide non-repudiation. The challenge for the future is to integrate all these value-added services in a single solution.

On the other hand, CORBA defines two security levels; Level 1 and Level 2. Level 1 allows a telematic service that is unaware of security to participate in a secure domain. It provides user authentication, authorisation via ACLs, data encryption and integrity, and optional non-repudiation. Level 2 requires telematic services to be security-aware, thus enforcing stronger versions of the security policies. Some CORBA vendors, such as Iona and Inprise, have provided a Secure Socket Layer (SSL) implementation of IIOP, called *Level 0*, that allows user authentication and data encryption.

Manageability
Microsoft Management Console (MMC) provides a unified GUI for managing MTS and MSMQ based components. Features include centralised configuration and administration, as well as remote deployment of components. As far as CORBA is concerned, both Inprise and Iona have sophisticated tools for centrally configuring and administering CORBA applications. Iona also allows CORBA applications to be centrally managed from any SNMP-compliant system management console (for example, OpenView).

Support for web-based telematic services
A telematic service is web-based when its front end (or presentation layer) is a web browser, and it does not necessarily mean that the telematic service is deployed over the Internet[5]. In this case, DCOM front ends in the form of ActiveX controls can execute within Internet Explorer and, via a plug-in, within Netscape Navigator. Furthermore, Microsoft's Active Server Page technology allows the seamless integration of both HTML and ActiveX clients with DCOM servers. It also allows DCOM services such as MTS and MSMQ to be used with web-based telematic services. On the other hand, Java-based front ends to CORBA telematic services can execute on all major browsers and platforms, and the Netscape Enterprise Server provides the Web Application Interface, which allows HTTP-based clients to communicate with CORBA servers.

Support for Internet/extranets
Telematic services that need to operate across the public Internet or the semi-public extranet are typically deployed across great distances and often through several firewalls. Such telematic services, when they involve transactions executed over the Internet, require additional security measures to ensure accuracy, confidentiality, and credibility.

DCOM provides two-factor authentication (through public certificate and smart cards), and Remote Data Service (RDS) support for Internet/extranet applications. Currently, there is limited support for SSL and its integration with NTLM security. However, this will be expanded to include SSL-to-Kerberos integration in Windows NT 5.0. On the other hand, several CORBA vendors support both SSL and two-factor authentication in their implementation, although these features are still immature.

Support for intranets
Telematic services which are limited inside an intranet are usually optimised for use within an organisation, have higher network bandwidth, and little or no firewall restrictions. Therefore, they can be built with more sophisticated front ends, both in terms of user interface and functionality.

DCOM-based telematic services can be built with sophisticated user interface and functional features, as every major development environment on Windows supports the rapid development of graphical DCOM applications. Furthermore, Active Data Objects allows the support of persistence, while MSMQ provides publish/subscribe capabilities. On the other hand, CORBA-based telematic services can be integrated with desktop tools using a DCOM/CORBA bridge available from several CORBA vendors. CORBA services such as event and persistence can also be used to add publish/subscribe and persistence features.

Comparison remarks
From Table 1, Table 2, and Table 3, which collectively constitute a decision framework supporting the selection between DCOM and CORBA, it is evident that DCOM and CORBA have similar architectures as both provide the infrastructure for supporting remote object activation and remote method invocation in a client-transparent way. They adopt a client/server based programming style and agree on the most fundamental aspects of their object models.

As far as the support of service engineering related properties is concerned, DCOM and CORBA differ in many respects. Most significantly, while DCOM provides a rich set of tools and technologies, it is essentially a Windows-only solution. Even though DCOM is available on other operating systems, key pieces such as MTS, MSMQ and MCS are not currently offered. Additionally, many of DCOM's value-added services are very new and are still maturing. On the other hand, CORBA's main strength, which is its availability from different

vendors, is also its biggest weakness. Since no vendor has a complete solution, integration issues are usually introduced when CORBA is used to build telematic services. For this reason, neither technology provides a complete solution for service engineering activities. However, both provide a solid infrastructure, and there are specific scenarios in which each excels over the other.

Another important remark that has to be stressed is that DCOM and CORBA have a comparable performance. However, DCOM's performance can be improved in certain circumstances by extending its remoting architecture (that is, the entire infrastructure that connects clients to server objects). This is possible because DCOM's remoting architecture has built-in extensibility. By supporting a mechanism called *custom marshalling*, DCOM allows a server object to bypass the standard remoting architecture and construct a custom one, optimised for a particular situation, without requiring source code modifications to the former[7].

In general, the proposed decision framework makes it evident that DCOM and CORBA have much in common and continue to converge in several aspects. However, each architecture has different origins, with consequent strengths and weaknesses.

Interoperability Between DCOM and CORBA

Because both DCOM and CORBA are being used in practice with considerable success, and because of the economic implications that result from this fact, it is unlikely that one platform will soon overwhelm the other. Therefore, interoperability between DCOM and CORBA is crucial[5].

Since CORBA 2.1, the interoperability of DCOM and CORBA is part of the CORBA specification. More specifically, bridges receive object invocations from a CORBA application, translate them into equivalent data structures for DCOM, and have the function call executed in the DCOM application. In a similar manner, DCOM clients can access CORBA objects through bridges[14].

Conclusions

Advances in distributed object platforms have been rapid in the past few years. These advances have been largely driven by increasing demand for efficient object creation, interaction, management, and distribution. Both DCOM and CORBA address these issues, and are increasingly being used to develop new telecommunications services as distributed object applications. However, further progress is expected, and as both technologies are still evolving, it is likely that in the near future they will converge in more areas.

DCOM is built on a proven desktop component architecture. DCOM-based applications are robust and perform well, while DCOM's integration into development languages and tools greatly simplifies application development. Furthermore, as Windows-based desktop systems exist in nearly all organisations today, these organisations will probably choose to use DCOM. Additionally, Microsoft services (MTS, MSMQ, and other mainframe integration tools) make DCOM an attractive infrastructure even for large organisations in enterprise-wide applications. However, DCOM is not a well-partitioned architecture and relies on a key optimisation for a single platform.

In contrast, CORBA has a more complete and well-defined architecture and provides a better solution for heterogeneous environments. It offers advantages in (value-added) services, platform and tool support, maturity, and overall architectural integrity. Furthermore, OMG IDL ensures an extensible architecture and support for both new and legacy applications. The disadvantages of CORBA are its complexity and variation in vendor implementations.

Therefore, DCOM is an effective solution for the development of telematic services in Windows-based environments, particularly by small organisations and departments. On the other side, a requirement for multi-platform support or for a choice with the least technological risk will drive an organisation towards a CORBA solution. However, such a decision will (should) be highly influenced by more general factors, such as the available information technology (IT) resources and skills, the IT structural characteristics and its relation to business units of the organisation, the desired level of standardisation, and the capability to adopt new technologies. It has to be noted that Java is also a candidate for the development of telematic services, but not the most prominent as a lot of the transparencies built on the core object models of OMA and COM/DCOM are yet to be defined for Java. The significance of Java is expected to rise rapidly as its standardisation proceeds[1].

It is envisaged that in the near future CORBA and COM/DCOM will interoperate via a standardised single two-way gateway specification (a bridge) between them[5]. However, with dissimilar object models, components will not collaborate as effectively across the gateway between the two environments as they can within each of them. For this reason, the service designer/developer will always consider a choice between CORBA and OLE/COM on the client, and between CORBA and COM/DCOM on the server. The decision framework proposed in this paper is expected to significantly assist the selection process.

Developing new telecommunications services using distributed object technology presents many challenges and alternatives. The correct choice is never at either end of the spectrum, but falls somewhere in the continuum that lies in between. Where it falls depends on both the user/customer (business) requirements that have to be satisfied, and the technical problems that have to be solved.

References

1 ADAMOPOULOS, D. X.; and PAPANDREOU, C. A. Distributed Processing Support for New Telecommunications Services. Proceedings of the IEEE/IEE International Conference on Telecommunications (ICT '98), Chalkidiki, Greece, 1998, Vol. III, pp. 306–310.

2 AIDAROUS, S.; and PLEVYAK, T. (Eds.), Telecommunications Network Management. IEEE Press, 1998.

3 BROWN, N.; and KINDEL, C. Distributed Component Object Model Protocol—DCOM/1.0. Microsoft Corporation, Nov. 1996.

4 CHUNG, P. E.; HUANG, Y.; and YAJNIK, S. DCOM and CORBA Side by Side, Step by Step, and Layer by Layer. Bell Laboratories, Murray Hill, NJ, 1998. http://www.bell-labs.com/~emerald/dcom_corba/Paper.html

5 DOLGICER, M.; and FISCHER, P. DCOM, Active X and CORBA Must Live Together. Application

Development Trends, April 1997, pp. 38–52.
6 GOPALAN, S. R. A Detailed Comparison of CORBA, DCOM and Java/RMI. Sept. 1998. http://www.execpc.com/~gopalan/misc/compare.html
7 GRIMES, R. Professional DCOM Programming. Wrox Press, 1997.
8 KRIEGER, D.; and ADLER, M. The Emergence of Distributed Component Platforms. IEEE Computer, **31**(3), March 1998, pp. 43–53.
9 LEWANDOWSKI, S. M. Frameworks for Component-Based Client/Server Computing. *ACM Computing Surveys*, March 1998, **30**(1), March 1998, pp. 1–27.
10 Microsoft Corporation and Digital Equipment Corporation. The Component Object Model Specification. Draft Version 0.9, Oct. 1995.
11 Object Management Group. The Common Object Request Broker: Architecture and Specification. Revision 2.0 July 1995 http://www.omg.org/library/o2indx.html
12 ORFALI, R.; HARKEY, D.; and EDWARDS, J. Instant CORBA. John Wiley & Sons, 1997.
13 PROZELLER, P. TINA and the Software Infrastructure of the Telecom Network of the Future. *Journal of Networks and Systems Management*, Dec. 1997, **5**, pp. 393–410.
14 REDLICH, J.-P.; SUZUKI, M.; and WEINSTEIN, S. Distributed Object Technology for Networking. *IEEE Communications Magazine*, Oct. 1998, **36**(10), pp. 100–111.
15 VINOSKI, S. CORBA: Integrating Diverse Applications Within Distributed Heterogeneous Environments. *IEEE Communications*, Feb. 1997, **14**(2), pp. 46–55.

Biographies

Dionisis X. Adamopoulos
University of Surrey

Dionisis X. Adamopoulos holds a degree in Computer Science from the Athens University of Economics and Business, and a Masters degree in Telematics with distinction from the department of Electronic and Electrical Engineering of the University of Surrey. Currently, he is involved in Ph.D. research at the Centre for Communication Systems Research (CCSR) of the University of Surrey. His research interests include service engineering, distributed multimedia, object-oriented analysis and design, telematic services, distributed object platforms, groupware, and telecommunications management.

Dr. Constantine A. Papandreou
Hellenic Telecommunications Organisation (OTE)

Dr. Constantine A. Papandreou holds a Dipl. Ing. degree as well as a postgraduate degree in Engineering Economics both from the Technical University of Munich. He also holds a Doctor degree in Telematics from the University of Munich. He has served since 1970 in the Hellenic Telecommunications Organisation (OTE) as an expert in telecommunications and informatics in various positions. Since 1992, Dr. Papandreou has been a Director in OTE. In parallel to these activities, for several years he has been teaching teleinformatics and information technology at the Athens University of Economics and Business, the University of Piraeus, and the Higher School of Telecommunications of OTE. He is the author of over 60 scientific papers pertaining to his research activity in the fields of telematics, information systems, telecommunications policy, education and training, multimedia, office automation, service engineering, etc.

Prof. George Pavlou
University of Surrey

Prof. George Pavlou received his Diploma in Electrical and Mechanical Engineering from the National Technical University of Athens, and his M.Sc. and Ph.D. in Computer Science, both from University College London. Over the past 12 years he has been undertaking and directing research in the areas of data communications and telecommunications with emphasis on performance evaluation, network and service control and management and the use of distributed object-oriented technologies in new telecommunications architectures. He has contributed to ISO, ITU-T, NMF/TMF, OMG and TINA standardisation work. He has published around 50 papers in international refereed conferences and journals and he is the co-author of two books. He is currently professor of information networking at the University of Surrey, School of Electrical Engineering and Information Technology, where he leads the activities of the networks research group.

Emmanuel Manolessos
IBM Hellas

Emmanuel Manolessos holds a degree in Computer Science from the Athens University of Economics and Business, and a Masters degree in Advanced Information Technology from the department of Computer Science of the Imperial College, University of London. Currently, he is a SAP consultant at IBM Hellas. His research interests include distributed-processing environments, interactive teletraining, enterprise resource planning, and management information systems.