Poseidon House
Castle Park
Cambridge CB3 0RD
United Kingdom

TELEPHONE: **Cambridge (01223) 515010**
INTERNATIONAL: **+44 1223 515010**
FAX: **+44 1223 359779**
E-MAIL: **apm@ansa.co.uk**

ANSA Phase III

Open Distributed Processing - the Solution to a Business Need

Andrew Herbert

Abstract

This is a slide presentation, designed to take about 90 minutes which sets out to explain the problem ANSA and ODP are trying to solve, why the solution requires an architecture and how iindustry offerings match up against that architecture.

APM.1055.01 **Approved** 17th October 1994
Briefing Note

Distribution:
Supersedes:
Superseded by:

Open Distributed Processing

the Solution to a Business Need

Andrew Herbert

ANSA Chief Architect

Technical Director
Architecture Projects Management Limited

Contents

The subject of this talk is diversity in IT systems and what if anything can be done to alleviate the problem. First we look at some of the key problems facing significant IT users today

- **Key Problems**

and then discuss some of the new and fashionable methods for dealing with these problems, finding out that one solution won't suit all.

- **Current industry offerings**

Then we shall look at what the requirements really are for overcoming diversity and the characteristics we want of a good solution

- **A Wish List**

We then explore a different way of approaching the problem - seeing it being one of architecture rather than product selection

- **An Architectural Solution**

And then lets understand how current and upcoming "standards" fit into the architecture.

- **Standards**

Then a look at the ISO standard based on the ANSA architecture - altough only time for a quick skim over the surface

- **ISO/CCITT Basic Reference Model for Open Distributed Processing**

And finally wrap up the main points.

- **A Conclusion**

Key Problems

- **Incompatible systems**

Because of technical diversity of hardware, operating systems (400 Unixes), comms, programming languages, databases, guis - explosion of combinations - compatibility = hard work

- **Protecting legacy systems**

They work - need them - no business case for replacement - but legacy systems. don't support integration

- **Scaling up**

geographic - time lags - time zones - span several administrations; network size - global reboots or updates of s/w not possible; requires evolution

- **Scaling down**

computing is moving out of mainframes into pcs and workstations, how to exploit tiny computers?

- **Rate of change of technology**

Technology changes annually - changing is painful, expensive, inconvenient - not changing leaves competitors ahead - if change some, how to integrate new with old

- **Responding to organizational change**

enterprise controls system / system controls enterprise? IT forces working style, inhibits change. Mergers etc => force IT to change, but IT can't change as fast as Enterprise can. Building societies example.

- **Manageability**

Commercial pressure => fastest way to update/fix s/w. Continual patching => "structural fatigue"; can become impossible to meet change requirements (can do but not in time)

Current Industry Offerings

Some of the industry's favourite techniques at this moment. Individually:

- ## Rightsizing

(Downsizing) Replace mainframe by UNIX workstation. Very fashionable, not always necessary. Why not retain mainframe as integral part of network? Good thing: forces review of whole system.

- ## Object Orientation

Risen in last 2-3 yrs. Way of thinking about system components, as black boxes. Divide and conquer - can help quick fix and diversity to some extent, properly applied.

- ## Client/Server

Tendency to think of hardware servers, talking about software here. Way of thinking about communication between system components. Server can be remote - over 20 yrs old, re-discovered - helps geographic spread. Different client interfaces to legacy applications and corporate information for different jobs.

- ## Open Systems

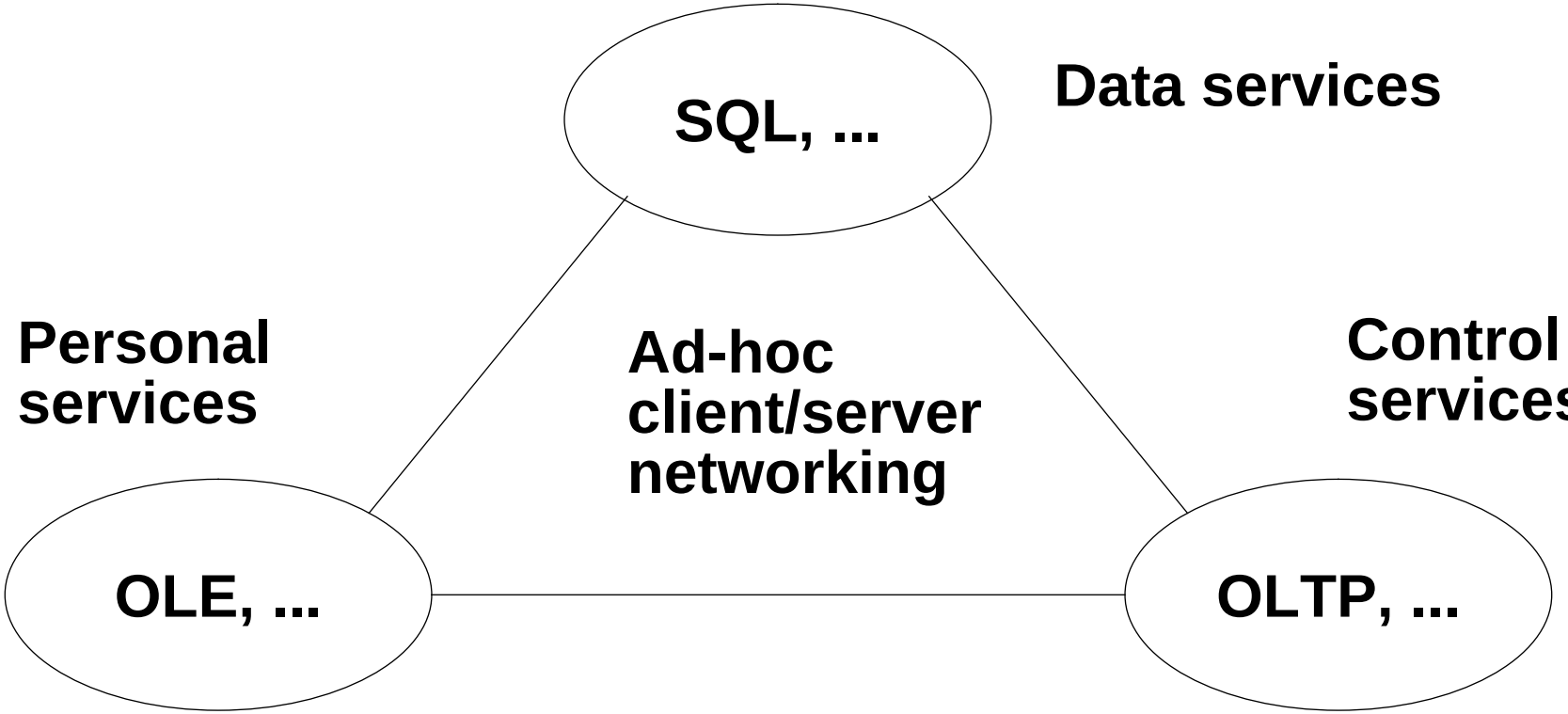
Anyone like to offer a definition? UNIX, RDBMS, GUIS; at Open System Show @ Olympia... windows NT ... 15 languages & 20 countries. Should be the Proprietary Open Systems Show. Point is: all above are ingredients. None is a solution by itself. Many suffer from over-hyping. Lots of confusion re: how to fit together.

- ## No single technology solution can win

Because of legacy issue; it will become out of date before it gets everywhere; what to do in the transition? Will the market allow one vendor (group) to take over?

A Service-Orientated View

We can divide up current approaches to distribution into three sectors, which have different requirements and therefore different solutions. They key is to think in terms of “services” rather than “components”.



Data Services

This is the database, centralized MIS culture trying to cope with right-sizing and data integration. It is rooted in a 1960s world of mainframes, WANs and IT departments isolated from user's needs, coming to grips with PCs and LANs

- ## Remote data access

OSI RDA, DB vendor client/SERVER systems; SQL everywhere. Hidden in most cases behind a 4GL or GUI

- ## Federated databases

Currently read-only federation mostly. Dealing with inconsistency is hard. Preserving autonomy of each datavbase is hard.

- ## Stored procedures

Putting applications in the database - good for integrity, but becomes a bottleneck and slow to change, OO SQL almost a programming language

- ## Object repositories

Getting the the information to manage itself. OMG Object Services. Should the repository be distributed? Naming and typing is hard if multiple administrations get involved.

Personal Services

A whole new set of services that have come in the wake of PCs. PCs are no longer toy computers in terms of MIPS, Mbs and networking, but solutions don't scale well because of the single isolated user legacy. The origins of this part of the industry in the back bedroom pushes it towards lightweight ad hoc solutions and a rapid turnover of new products and releases. Its the area of the market that has most growth potential and every vendor is fighting to own it

- **Personal productivity services**

Cross-linking applications. Microsoft OLE. GUI-based CLIENTs to remote servers, e.g. Executive Information Systems

- **Group productivity services**

Groupware. Interconnecting users. E.g. distributed diaries. All based on objects, leading to new wave of OS with low-level support for objects and object linking across networks- Microsoft CAIRO, Taligent PINK, Unix vendors COSE

- **Mobile computers**

Upcoming challenge. Laptops. Palmtops. What to do when laptop reconnects?

Control Services

This is the networking culture that has grown up around super-minis and fast networks. Its mentality is all about reliably getting messages from A to B.

- **Online transaction processing**

Lots of products. Each do one job very well. Need to be a system expert to use them. Open standards exist OSI TP X/Open XA.

- **Workflow**

Workflow idea - derive system from specification of the job it does. Applies to clerical activities, financial trading, edi,, computer-aided manufacturing etc. Coming to market now.

- **command and control**

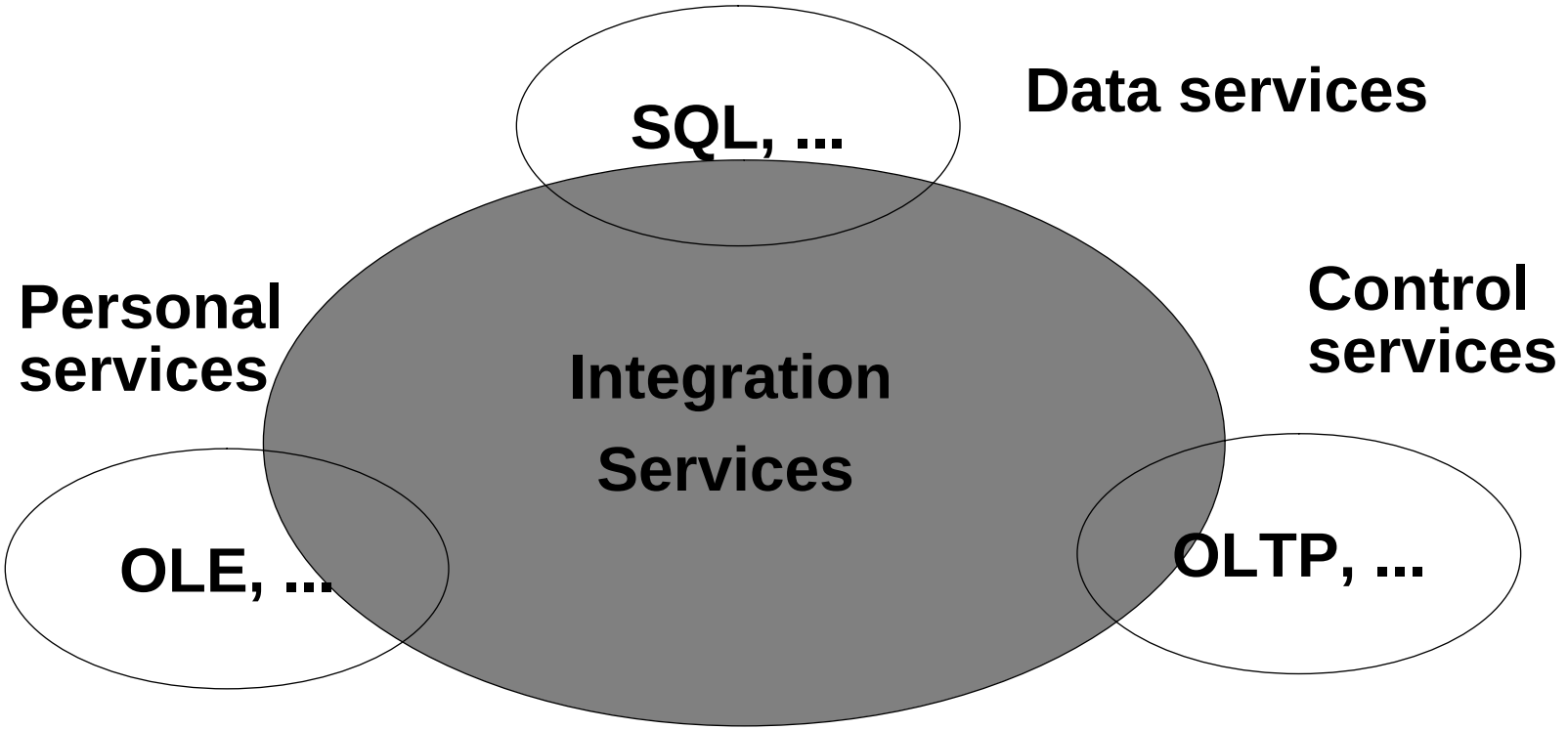
Using computers to pull together standalone autonomous systems to meet a common goal. Robust messaging and EDI are main elements.

- **Intelligent networking**

All about telecomunications bearers of-loading services to other parties - letting other people run programs on your system is a major challenge

All of these have stringent security, performance and reliability constraints, so solutions are heavily engineered, which makes then inflexible.

Linking Services Together



Integration will not only have to link the other kinds of services, but be able to penetrate their domains

A Wish List

A truly open solution should:

- **Integrate products from many vendors**

Exploit innovation, price / performance trade-offs

- **Span application domains**

Information aturally flows from one dept to another; nees to do so economically and automatically

- **Hide system boundaries**

Size should only affect performance, but not functionality

- **Enable inter-organisation computing**

Controlled federation - don't want to expose everything; have to resolve policy conflicts

- **Preserve existing investments**

Commercially sensible, requires an environment which facilitates evolutionary change

- **Match IT style to Enterprise requirements**

Enterprise should drive the system, IT should respond to (facilitate) change. Can't rely on a single administrator

- **Allow rapid, low-risk adoption of new technology**

No-one wants to be first, no-one can risk being last....

- **Be manageable**

You need to know what's out there and who's using it, ...

Reaching the Solution

- **Specify interfaces using application concepts**

then they will be durable against technical change and comprehensible to applications programmers. The technical term here is abstraction - this is a good thing in computing; consider 4GLs which abstracted away from DB access methods and COBOL, structured programmed which weaned us off assembler, ...

- **Define a standard model for distributed programming**

Agree what programmers can and can't do - semantics are important, not syntax; limit the number of options to increase the level of abstraction and enhance portability

- **Use tools to automatically generate the technology interfaces**

because its cheaper than writing it and it doesn't have to be maintained; you can even afford to throw it away! Follow the trend towards every more "declarative' input (i.e. say what you want, not how to do it)

- **Define structures for integration**
  - **management and monitoring**

What is going to be controllable in the system, where is it controlled from?

  - **trading and federation**

How is my system linked to yours - what services do we provide to one other - what "adaptors" do we need?

  - **transparency**

what is going to be hidden (by automation)?

- **Define models for common application domains**

so we can talk to each other - to get here we need to start from some common "describing languages"

Requirements on the Solution

An approach to system specification, design & construction is needed which is:

- **Easy to use, easy to understand**

Reason obvious: no-one will use if hard to use, or if they can't understand it

- **Self-consistent & coherent**

If not, dangerous, as the approach itself could introduce inconsistencies in design; it had better all hang together; it approach doesn't, systems won't

- **Unrestrictive**

Not prevent you, for proprietary reasons, from doing what you need to; on the other hand strives to remove gratuitous differences between systems.

- **Generic**

A minimal set of principles which apply to all applications, all systems, all organizations

- **An Architecture not a shopping list**

Users will buy from vendors who support the architecture; users will pressure vendors to interconnect technologies with or without an architecture to do so - having an architecture will make this easier for vendors and give users more freedom to choose - having an architecture does not prevent vendors inventing new technology and/or new services

What is an Architecture?

- **Architecture is not a block diagram, but rather ....**

a more general notion of a set of standard ways of doing things to simplify integrationand remove unnecessary differences between systems: similar to the notion of “production engineering” in manufacturing. Ultimately should link into user’s application design process.

- **A set of basic components**

For us, the services we need to do integration; Services defined by their interfaces, not their technology. Goal is smallest set that can be combined to meet all requirements and hence depends upon abstraction to reduce the set of choices to the smallest NECESSARY set.

- **Design rules**

Ways to structure and combine systems so as not to fall foul of key problems like scaling, and other traps.

- **Recipes**

Ways to build components into more complex subsystems which behave in the way you want - a cookbook for designers

- **Guidelines**

When to break the rules and what the penalties are - optimisations - trade-offs

- **.... which follow a set of architectural principles**

ANSA

- **ANSA is such an Architecture**

Not new; work started 1984.

- **Practical and proven**

Not just theory; software built using ANSA since 1988. Original proof of concept (ANSAware), now a set of tools and components for enabling building of distributed systems. Paved the way to e.g. DCE, CORBA for sponsors.

- **Vendor-neutral**

ANSA makes no reference to particular products. Independence is guarantee not only of universal applicability now, but also in future.

- **Collaboratively developed**

by a group of 20 computing & telecomms cos. including DEC, HP, ICL, BT, Northern Telecom, France Telecomm & Siemens - leaves few issues untouched and brings in a wide range of requirements.

- **Backed by authority of ISO standards (ODP)**

Can build without stds; stds => guarantees. ODP - Basic Reference Model for Open Distributed Processing - emergent standard, ISO 10746 - 80% taken from ANSA.

- **Demonstrated capabilities now turning up in industry standards (DCE, CORBA)**

ANSA principles - in one slide

- **Distributed systems have different properties to centralised systems**

Differences are quite striking, and system design and implementation. techniques need to take this into account. Techniques and ideas based on central systems need revising to make appropriate. for distributed systems

- **Hide unnecessary complexity from programmers, where appropriate**

No doubt distribution more complex - more errors, more information about system to manage. Ought to make them harder to use - but it needn't if all unneccessary complexity hidden. A question of finding good abstractions (i.e. the right things to want)..

- **Different users need different solutions**

Do not expect one way of doing things will suit all uses and all technologies. Allow maximum freedom to vary and show how to cross boundaries where needs mismatch.

- **Adopt a simple but generic view of system components and their interaction**

Need a careful balance between simplicity and expressive power; don't want more complex than it needs to be so that conversion at boundaries is straight forward, but also not so simple that using it is difficult and long-winded; generic enough to apply to any component, whatever it does, or to any interaction. If you can do it for one component, it should work for all components.

Reversed Assumptions

- **Review, and reverse, many traditional system design assumptions:**

Distributed systems design is different: the are additional problems to grapple with, and new benefits to consider

TRADITIONAL	REVERSED
Local	Remote
Sequential	Concurrent
Single Environment	Diverse Environment
Fixed Location	Mobile
Single Copy	Multiple Copies
Synchronous	Asynchronous
Direct	Indirect
Shared	Separate
Global	Context Relative
Complete Failures	Partial Failures
Early Binding	Late Binding

Different Applications, Different Needs

There isn't going to be a single response to the wish list, because we are in a world of some very fundamental trade-offs, including.....

- **ABSTRACTION versus SPECIALIZATION**

The more you hide, the less chance there is to tinkerl

- **CONSISTENCY versus AVAILABILITY**

Availability means copies, increases risk of inconsistency

- **AUTONOMY versus UNIFORMITY**

Autonomy gives more freedom, but leads to differences which increases complexity

- **SECURITY versus CONVENIENCE**

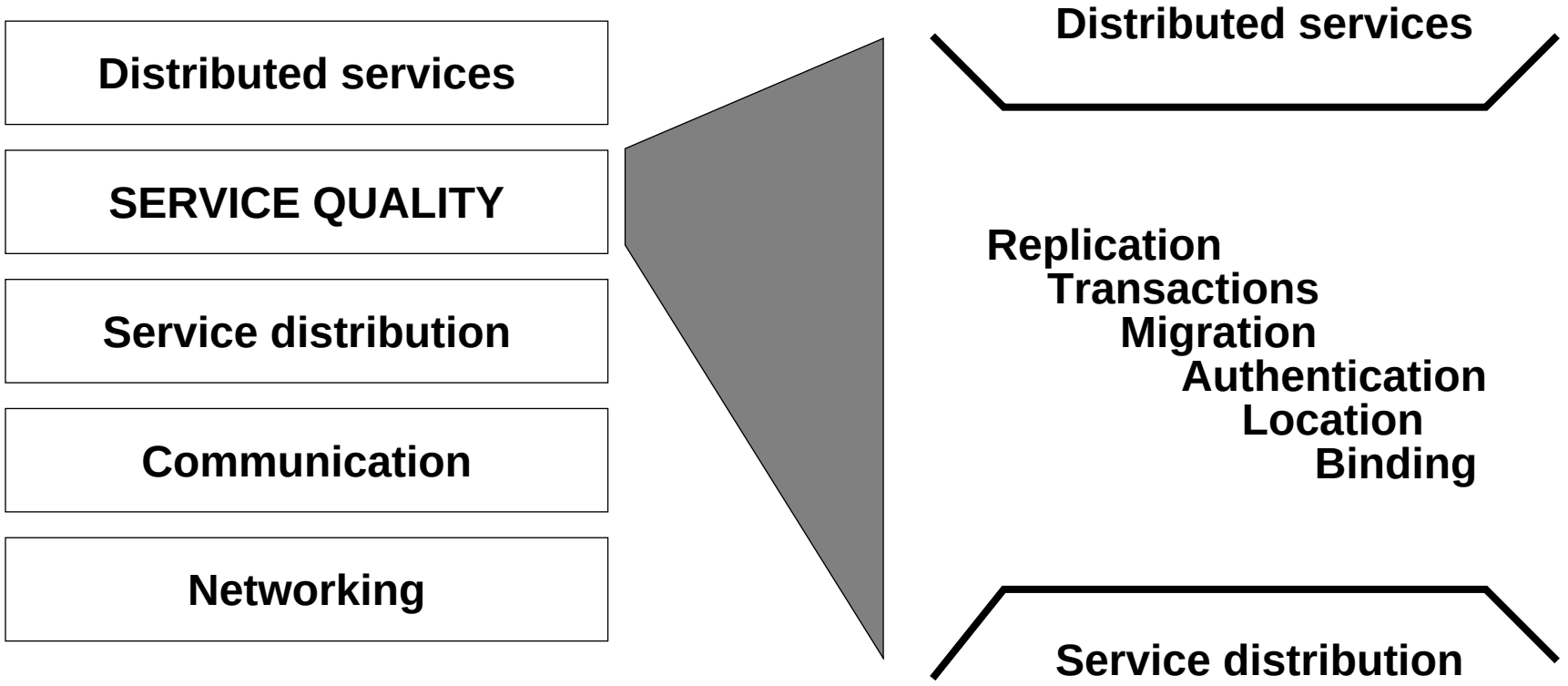
Security makes things harder to do

- **any many other trade-offs**

Means that one size does not fit all.

Structure of a Distributed System

This is a simple diagram to explore the engineering aspects of building distributed systems and then looking at relevant standards.



Selective Transparency

Service quality is about hiding irrelevant complexity - i.e. transparency:

- **Location**

don't need to know where it is to use it

- **Access**

don't need to know how it works to use it

- **Migration**

it can move while your using it, to balance loads or reduce latency

- **Replication**

there may be multiple copies for reliability and/or availability

- **Resourcing**

it gets resources when you use it and gives them back when you've finished

- **Partial Failure**

it always gets to a consistent state

- **Federation**

you don't have to have the same administrator to use it

The real productivity benefits come from automating transparency provision

Where are the Standards?

So, how much of engineering solution is available and standardized?

- **ISO / CCITT**

treaty-based organizations, make specifications by consensus, very slowly, but good at covering all the issues

  - **OSI TP service and protocol**
  - **OSI Management information model and protocols (GDMO, CMIS/CMIP)**
  - **OSI Remote Data Access, SQL**
  - **Open Systems Environment (POSIX)**
  - **Basic Reference Model for Open Distributed Processing**

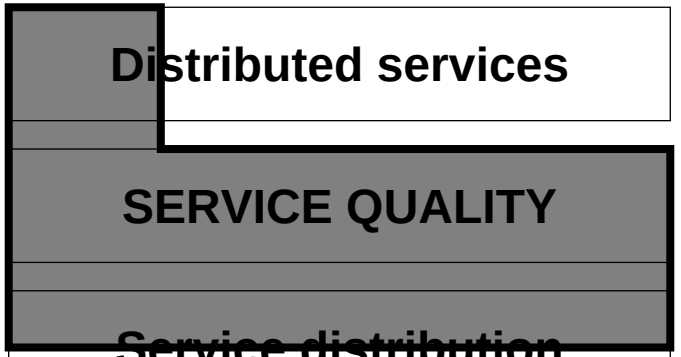
- **Industry standards**

Industry funded. Some agree requirements, some make specifications, some make technology

  - **OSF Distributed Computing Environment, Distributed Management Environment**
  - **Unix Intrnational Atlas**
  - **OMG Combined Object Request Broker Architecture, Object Services**
  - **X/Open XA**

- **De facto standards**
  - **MicroSoft Object Linking and Embedding (OLE), NT, ...**
  - **COSE**

RM-ODP



Communication

Networking

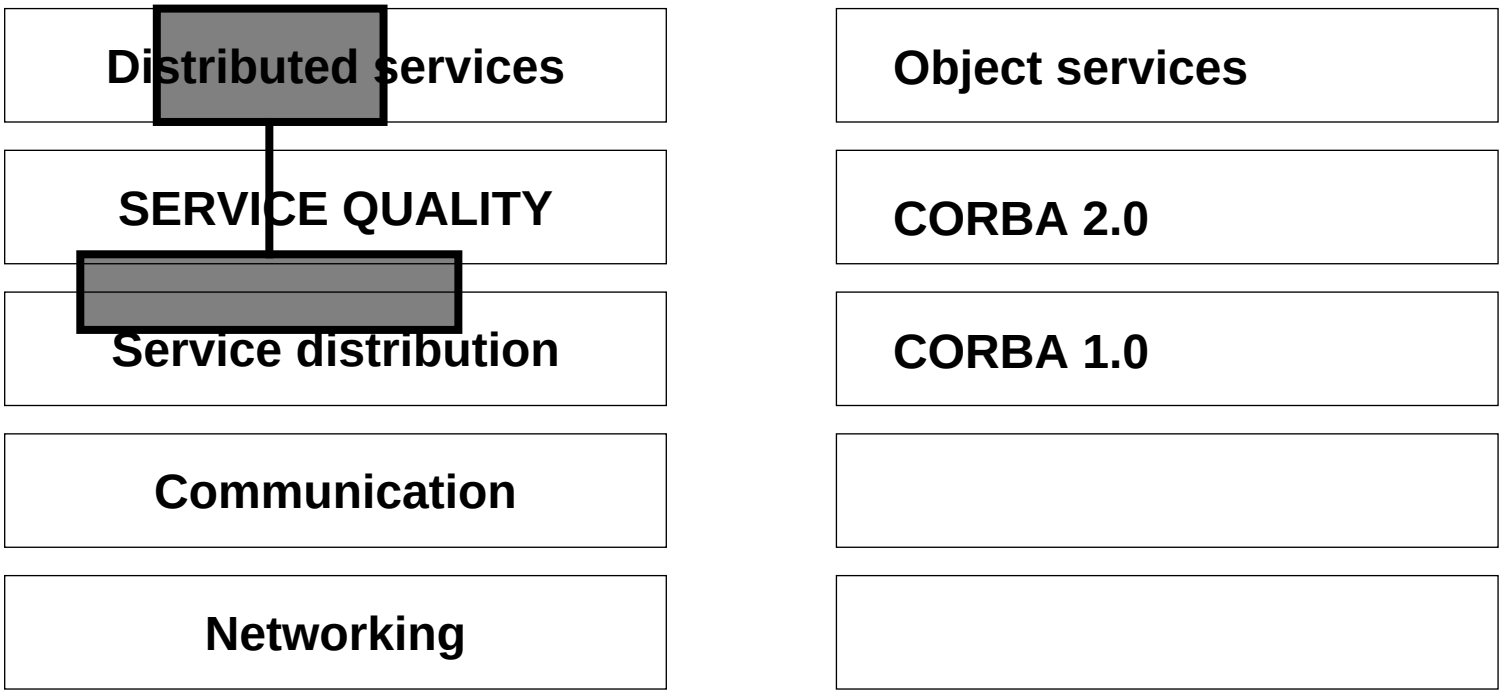
Trading

Transparency functions

Nucleus

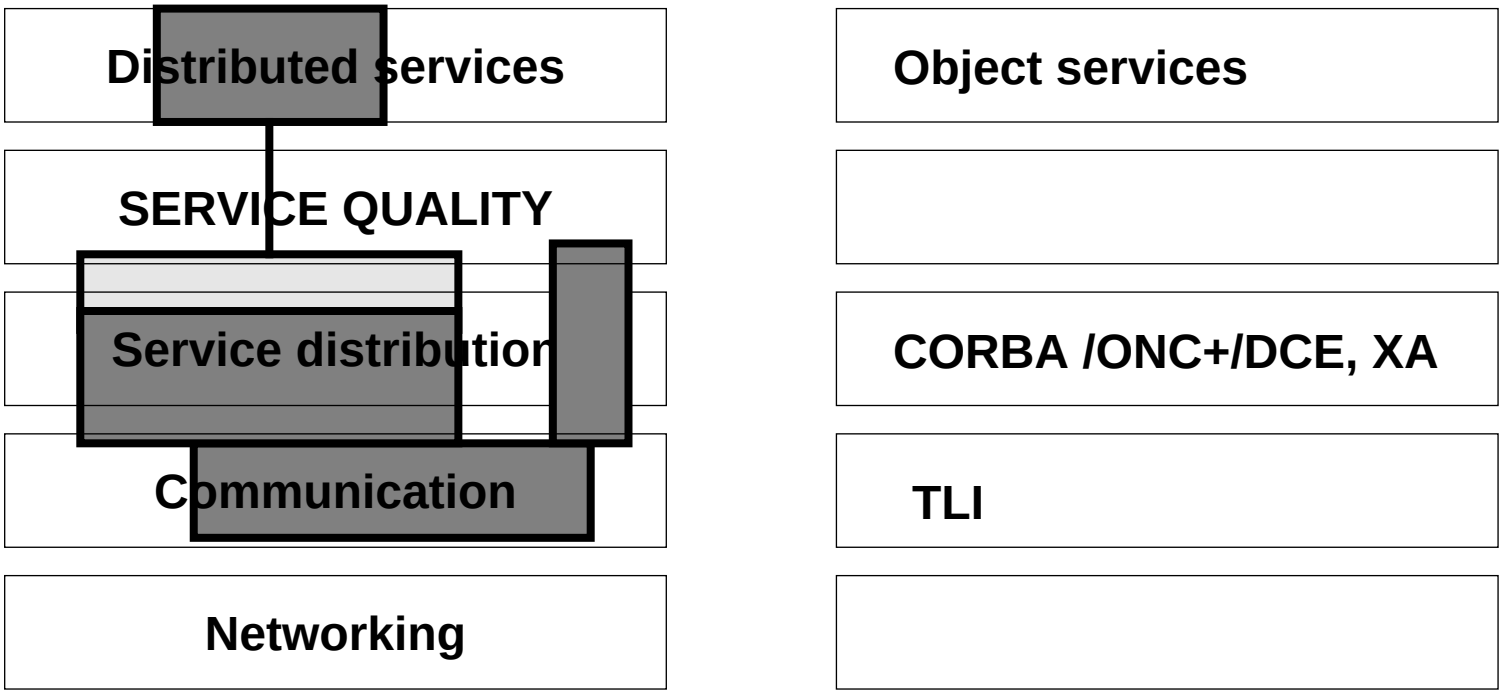

ANSA

OMG


Distributed services
SERVICE QUALITY
Service distribution
Communication
Networking
Object services
CORBA 2.0
CORBA 1.0

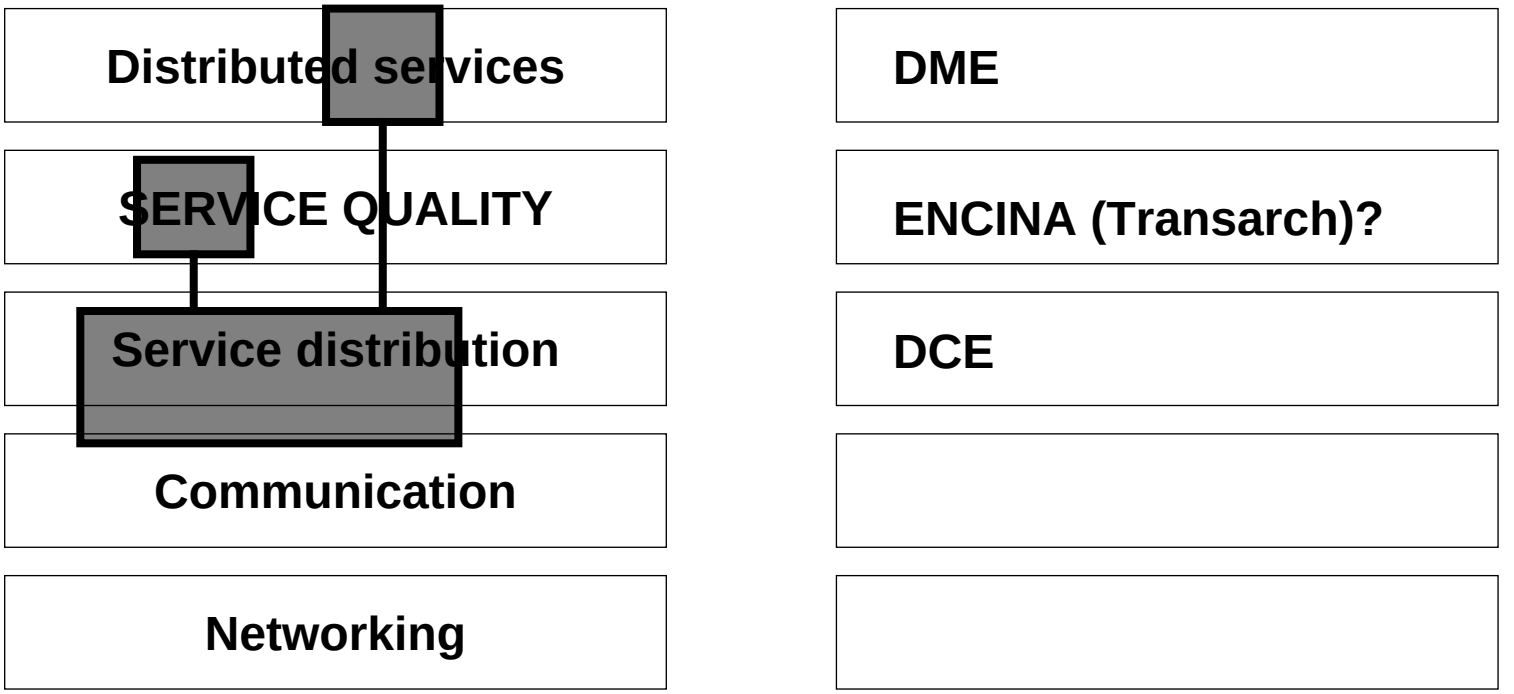

ANSA

UI/ATLAS


Distributed services
SERVICE QUALITY
Service distribution
Communication
Networking
Object services
CORBA /ONC+/DCE, XA
TLI

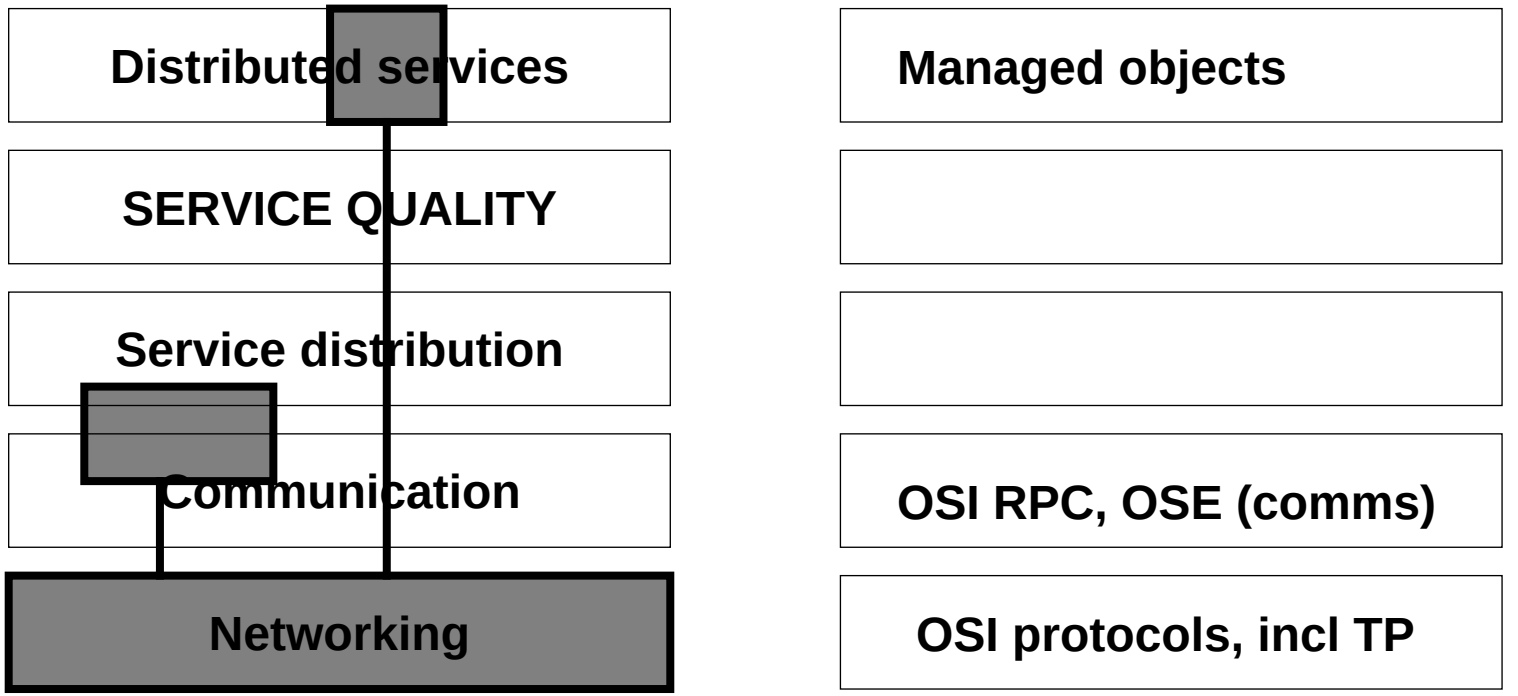

ANSA

OSF


Distributed services
SERVICE QUALITY
Service distribution
Communication
Networking
DME
ENCINA (Transarch)?
DCE


ANSA

OSI & OSE


Distributed services
SERVICE QUALITY
Service distribution
Communication
Networking
Managed objects
OSI RPC, OSE (comms)
OSI protocols, incl TP

Delivery

How will the standards be delivered across the many platforms we want to use?

- **PC's**
  - **as part of the application software**
  - **in the operating system (NT, Cairo, Pink)**
  - **=> lightweight client-oriented implementations**
- **Traditional Unix**
  - **as a library over the kernel**
  - **=> duplication of functions and performance penalties**
- **Microkernel Unix**
  - **service distribution as part of the operating system (Chorus)**
  - **with library for service quality (Chorus/COOL)**
- **Mainframe**
  - **built-in to the operating system**
  - **accessd via a library**
  - **via a front-end system**
  - **=> heavyweight, inflexible implementations**

Analysis

- **Lots of overlapping efforts**

And we've left many other important names off the list.....Even if we included them there'd still be gaps left on the block diagrams

- **Reflects different cultures and technology domains**

Which reinforces the point about not expecting a single product solution, or even convergence on a single interface....

- **All share open systems as a goal**

And who can define what "open" means any more.

- **All converging on service-oriented view**

Often expressed as "object-orientation"; o-o is (probably) the best technology for making services easily

- **ODP Reference Model spans the area where new technology is required**

And therefore gives us a framework for understanding how to link together the available technologies and how to plug the gaps.

ISO/CCITT Reference Model for Open Distributed Processing

This is necessarily a very high level view.....

Separating Concerns:

The RM-ODP gives five ways of talking about the system, from a different perspective, none more important than the other.Don't regard them as design stages. Design stages will use all viewpoints simultaneously but at different levels of granularity.

- **Five viewpoints, derived from the ANSA projections**
  - **Enterprise**

Concerned with how IT system fits Enterprise it serves - describes objectives in terms of boundaries, policies, roles, responsibilities, contracts etc.

  - **Information**

Structure and meaning of information flow around the system; constraints on validity of information

  - **Computational**

The functional components, how they interact, what they do and the way they do them

  - **Engineering**

Block diagram of the infrastructure supporting the system, transparency provision, interceptors/gateways/adaptors

  - **Technology**

The conformance requirements on components and interfaces; which standards, which products, where.

The ODP and ANSA Concept of “Service”

Services:

- **A conceptual view of an application or system component**

in trms of the contribution it makes to the overall system goal

- **Anything can be an ANSA service, even a legacy system**
- **Components (objects) are providers and users of services (client/server)**

Services need not equate with servers. Banking is a service spread over many servers. Customer Accounts are individual services within a Bank Branch Server, etc

- **Service associated with an interface**

Ehat lies behind the interface is no concern to the client. E.g. Electrical socket - service is electrical power. Don’t care how it gets to me, who makes it, where it comes from. However I do want to specifiy 240V, 50Hz, no blips

- **Quality of Service**

What guarantees accompany the functionality - e.g. reliability,availability, latency, accuracy, cost, ...

Trading:

- **Service users find service providers via Trading Services**

Just another services - which lets servers advertise, and clients gain access.

- **Service references are the currency of the system**

A reference is all you need to use a service. WIth the reference concept you can describe any interface abstractly.

Conclusion

- **A new approach to IT systems is both possible and desirable**

Approach exists already, has done for some time. Avoiding key problems, getting to state described on wish list - a desirable situation to be in.

- **Applying ANSA / ODP offers you an evolutionary route**

Lets you write systems in a style which is appropriate for a distributed environment, taking advantage of new freedoms and capabilities, whilst avoiding many of the traditional restrictions.

- **You can build on legacy systems, innovatively, with low risk**

Both the software and the hardware can be incrementally incorporated into an ODP system without undue difficulty. New services can be built, old ones modified, in a timely and evolutionary manner. Never throw away systems until you're ready to. Slogan here is "evolution not revolution".

- **Technologies supporting the architecture are coming on stream**

- **Many enterprises have already benefitted from applying ANSA**