Poseidon House
Castle Park
Cambridge CB3 0RD
United Kingdom

TELEPHONE: **Cambridge (0223) 323010**
INTERNATIONAL: **+44 223 323010**
FAX: **+44 223 359779**
E-MAIL: **apm@ansa.co.uk**

ANSA Phase III

ANSA Phase III - ANSAWorks Presentation

Andrew Herbert

Abstract

A 40 minute presentation outlining the scope of ANSA Phase III. Introduces the motivation for using distributed computing technology to achieve applications integration, the concept of “Management Engines”, the main areas of focus and work in progress.

APM.1183.00.01 **Draft** 29 March 1994

External Paper

Distribution:
Supersedes:
Superseded by:

ANSA

Phase III - Update

Andrew Herbert (Introduction)

Dave Otway (Performance)

John Warne (Dependability)

Rob van der Linden (Federation & Tools)

What is the ANSA Workprogramme?

- A collaborative industry effort to advance distributed systems technology
- Based on a shared architectural vision (ANSA)
- Vendor neutral
- Contributes to standards (ISO/ITU ODP, OMG)
- Produces advanced technology prototypes (ANSAware)
- Enables its sponsors to deliver more effective products and services
- Provides a technical resource for the sponsors to use
- Strong focus on computer and telecommunications service integration

Distributed Computing

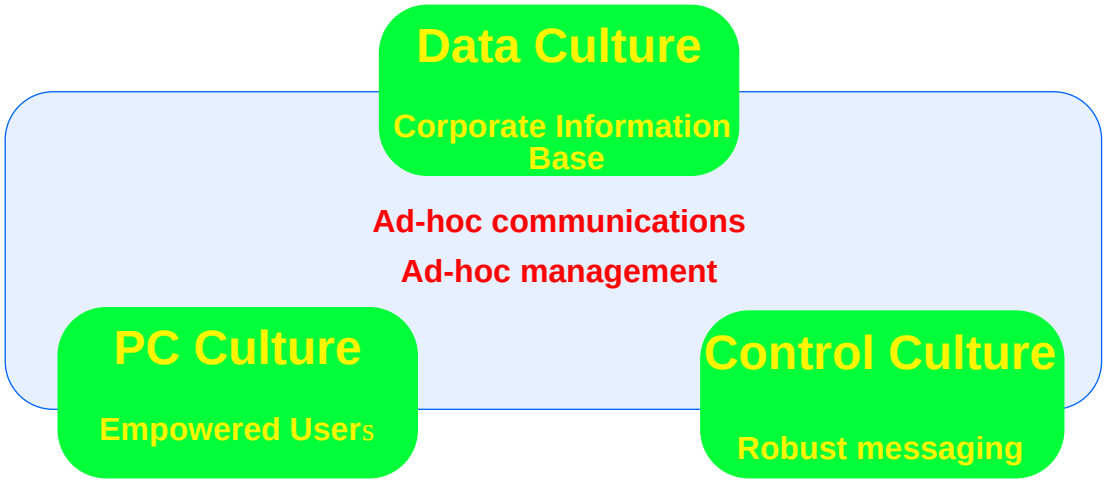
What are the upcoming issues?

Distributed systems have been an end in their own right, but that picture is changing

We need to recognize a wider context than RPC-based distributed computing

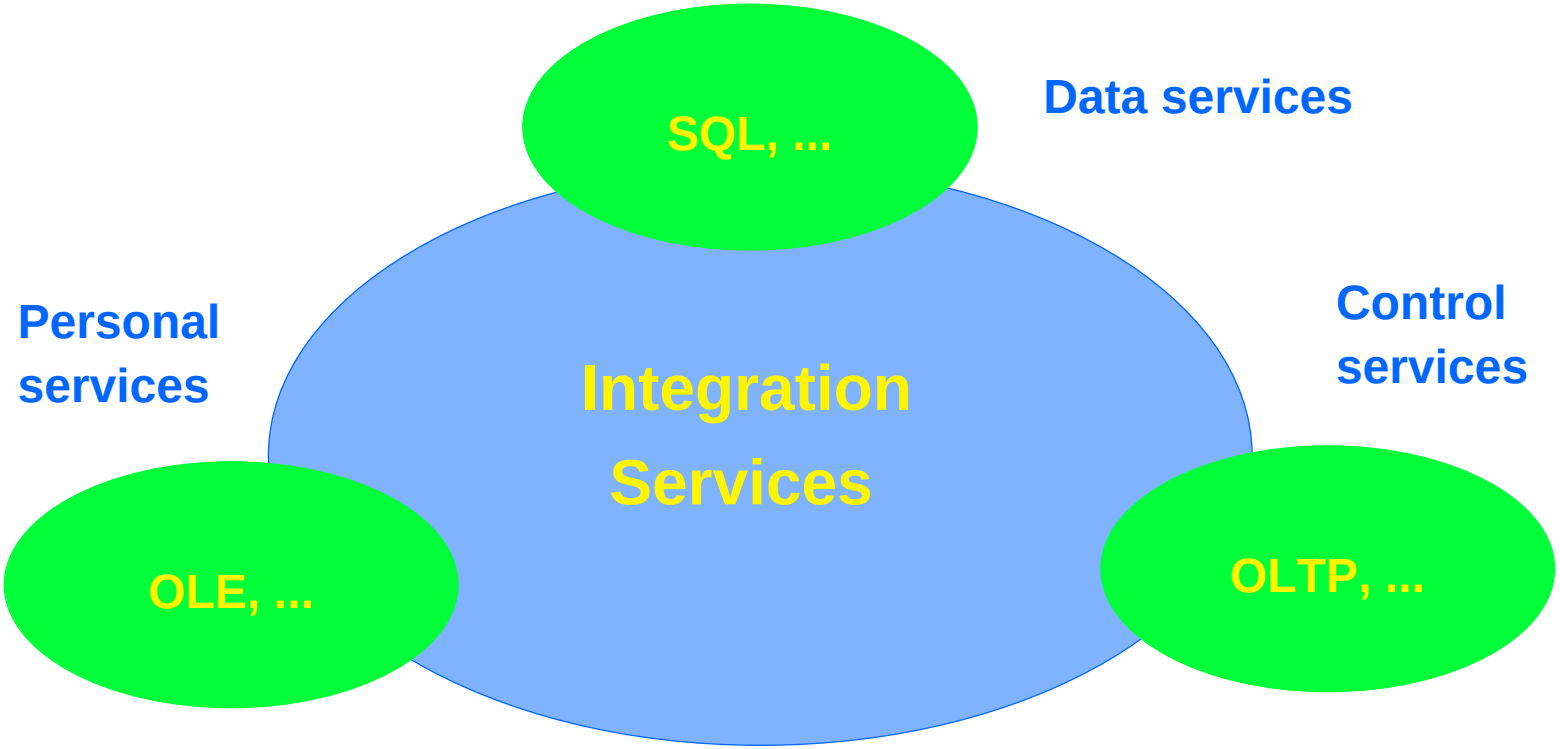
Distributed Applications Today

- Three important and quite separate distributed applications cultures exist



- Each has its own means of application distribution and integration
- They meet different needs - no single one will absorb the others
- Legacy of existing technology choices and applications will persist - RPC and object solutions won't displace them

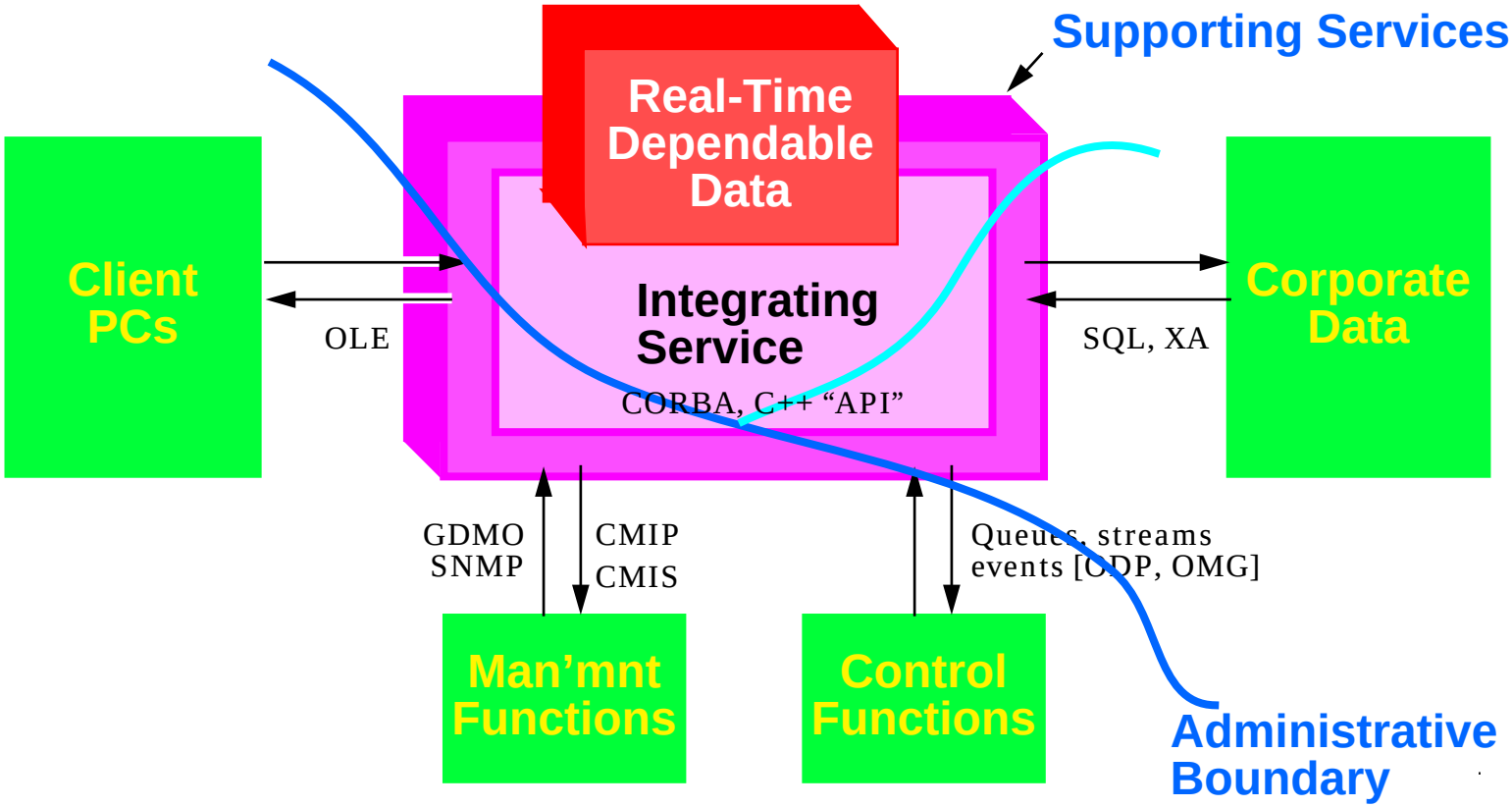
Distributed Computing = Integration Services = Interoperability and Management



The Generic Integration Wish List

- **Integrate products from many vendors** - exploit innovation, price/ performance trade-offs
- **Span application domains**- enable information to flow between departments easily
- **Hide system boundaries** - size sould only affect perormance, not functionality
- **Protect enterprise boundaries** - preserve autonomy and enable flexibility
- **Enable inter-organisation computing** - controlled federation - doing business doesn't require a merger
- **Preserve existing investments** - enable evolutionary change
- **Match IT style to Enterprise requirements** - make the business drive the system, not the other way round
- **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,....

Support for Integration Services - Management Engines



Management Engines

- A management engine provides high level supervisory control
- A management engine is built using distributed object computing
- A management engine must bring structural integrity to the systems it manages
- The other cultures have defined standards that give a narrow point of access
- The goal is to provide the library of components and the tools for building and evolving management engines
  - presumes means to relate system requirements to component properties, and
  - an architecture for the components

Building Management Engines - using ANSA Principles

- **Specify systems using application concepts**
  - good abstractions
  - minimise programmer book-keeping
- **Define a robust, high level model for distributed programming**
  - use objects for encapsulation / separation
- **Use tools to automatically generate the engineering detail**
  - generalize the idea of an *RPC stub generator*
- **Define structures for integration**
  - cradle to grave object management and monitoring
  - as much checking as possible, as early as possible
  - decentralized management - trading and federation
  - selective transparency over engineering detail

Distributed Programming

- **Distributed applications programmers must review and often reverse traditional design assumptions**

TRADITIONAL	REVERSED
Local	Remote
Sequential	Concurrent
Single environment	Diverse environment
Fixed location	Mobile
Single copy	Many copies
Synchronous	Asynchronous
Shared	Separate
Global	Context relative
Early binding	Late binding

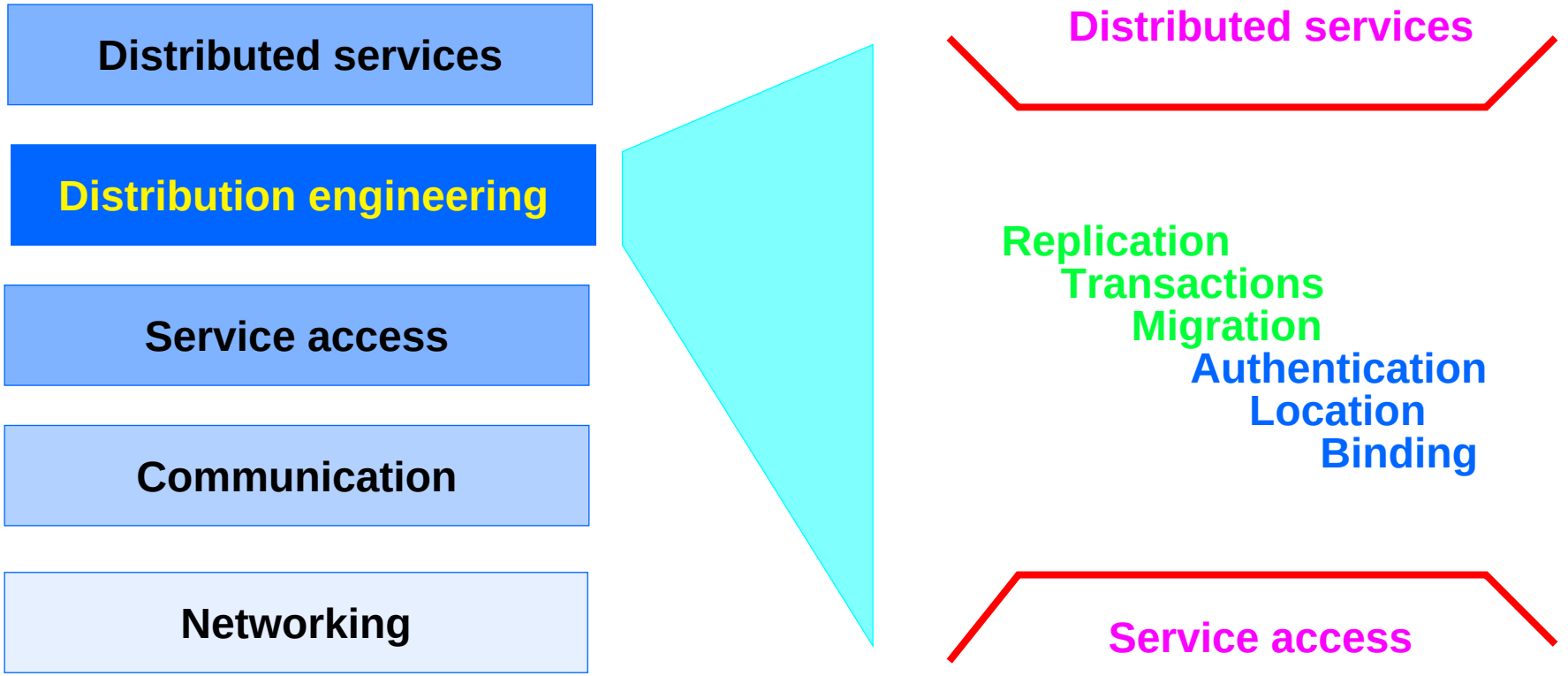
Distributed Object Computing

- Object is a unit of modularity for design, configuration, protection
- Object provide services
- Each service provided at an interface
  - operation interfaces contain “methods”
  - stream interfaces and signal interfaces for general communication
- Interfaces are access and relocation transparent
- References to interfaces can be sent as arguments and results
- Objects manage themselves, objects secure themselves
  - using supporting services

Distributed System Engineering

- **Distributed systems engineering is all about trade-offs**
  - **ABSTRACTION versus SPECIALIZATION - the more you hide, the less control you have**
  - **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**
- **Therefore we need a kit of engineering parts and automated transparency tools for linking application code to engineering code**

Engineering Structure of a Distributed System



Trading and Federation

- **Traders are the kingpin of system management**
  - trader is a reposiory of service offers
  - offer = interface reference + type + properties
  - server (server's agent) exports, client (client's agent) imports
  - types must match for interaction safety, properties steer selection process
- **Offers can be for applications, factories, distribution services and wrapped system resources or legacy software**
- **In a large system there will be many traders**
  - optimized to different needs
  - interconnected to form a trading federation
- **No single administrator is in control**
  - local autonomy in choice of names and setting trading policy
  - transparency requires context relative naming and interception at domain boundaries

Phase III Areas of Focus

Applications Developers

SYSTEM MANAGEMENT

TRADING

FEDERATION

PROGRAMMING MODELS

CONSISTENCY WORKFLOW TIME, EVENTS

REPLICATION ATOMICITY QoS

AUTOMATED ENGINEERING

groups

atomic infrastructure

message delivery guarantees

scheduling guarantees

CORBA
DCE

real-time kernels
high speed networks

Performance **Dependability** **Federations & Tools**