



---

**Poseidon House  
Castle Park  
Cambridge CB3 0RD  
United Kingdom**

TELEPHONE:  
INTERNATIONAL:  
FAX:  
E-MAIL:

**Cambridge (01223) 515010  
+44 1223 515010  
+44 1223 359779  
apm@ansa.co.uk**

---

**APM**

## **Building a Better Legacy**

**Andrew Herbert**

### **Abstract**

This is a technical/marketing presentation outlining the ANSA Architecture as a vehicle for systems integration.

---

APM.1513.02

**Approved**  
External Paper

24th July 1995

---

**Distribution:**  
**Supersedes:**  
**Superseded by:**

Copyright © 1995 Architecture Projects Management Limited  
The copyright is held on behalf of the sponsors for the time being of the ANSA Workprogramme.





## **Building a Better Legacy**

# **Using Distributed Object Technology for Effective Systems Integration**

**Andrew Herbert  
ANSA Chief Architect**



## The Open Systems Movement needs.....

- a credible future open systems vision
  - rationalizing the past isn't enough
- building up from current technologies
  - to maximize return on investment
- intercepting new capabilities
  - to ensure the market grows
- with an order of magnitude reduction in systems integration cost



## How ANSA can help

- **An architecture for Open Systems based on**
  - trading and federation
  - custom infrastructure
  - abstract and automate
  - modular engineering
  - controlled interoperability
  - one size does not fit all
  - tools replace APIs
  - open internal interfaces
- **developed by the ANSA consortium**
  - proved in real products and systems
- **consistent with key industry standards**
- **with a firm grip on the future**

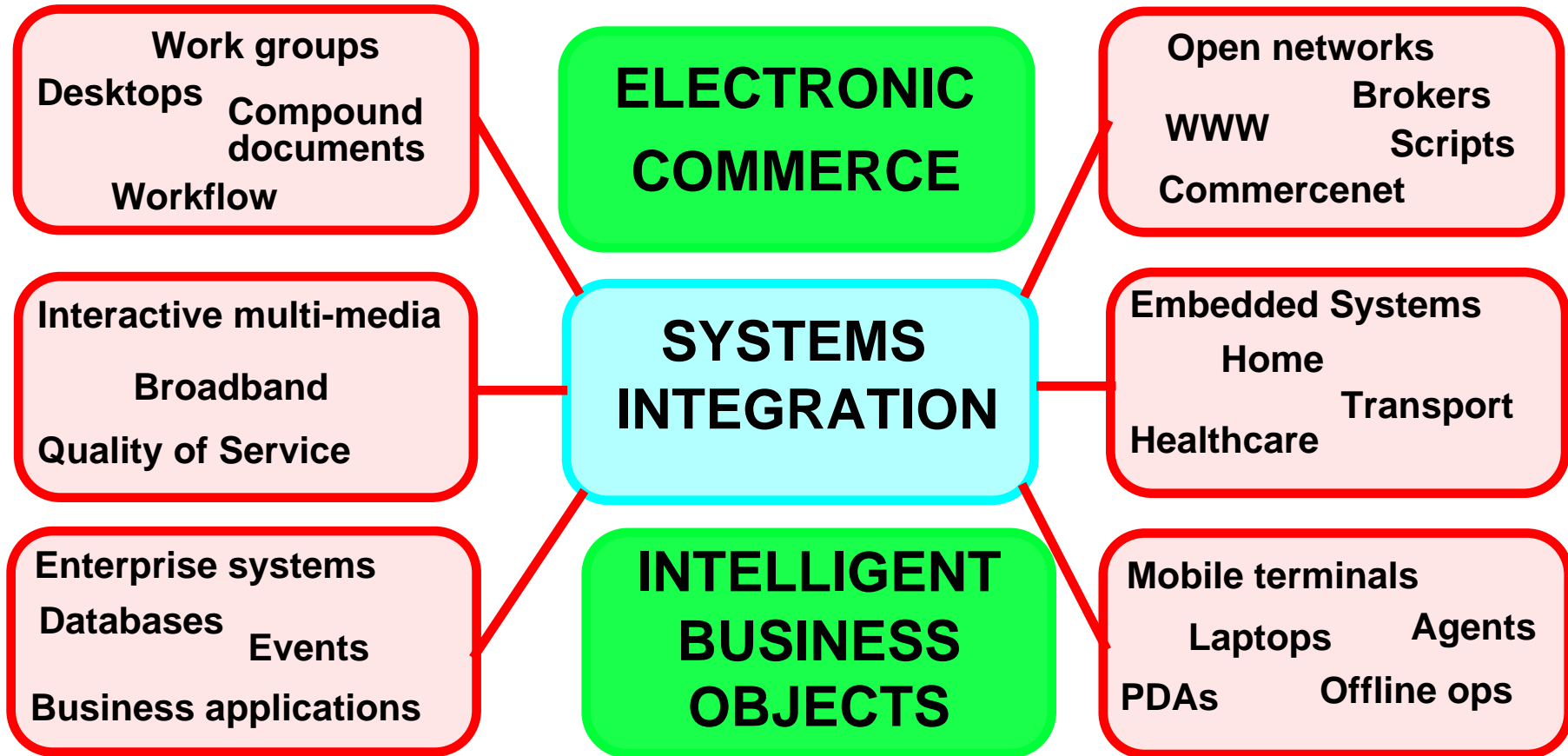


## The ANSA Process

- **ANSA Consortium**
  - **Founded in 1985**
- **Representative membership**
  - *computer vendors*                      *telecoms manufacturers and operators*
  - *systems integrators*                      *end users*
- **Shared core team**
  - **research, scenarios, animations, develop prototypes**
  - **technology transfer**                      *secondment, projects, training, consultancy*
  - **influence key standards**                      *OSF, OMG, ISO/ITU ODP, TINA*

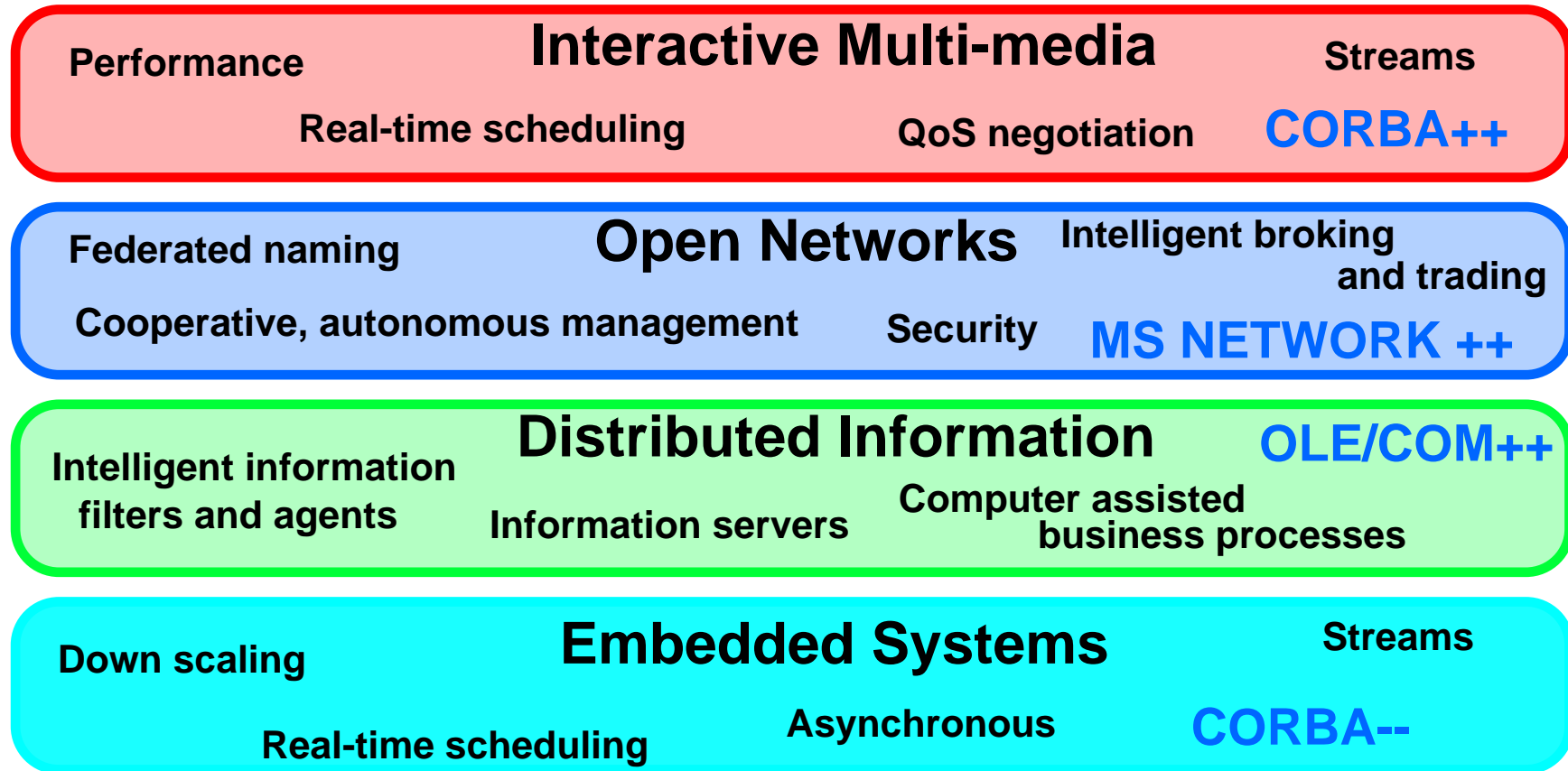


## ANSA Vision



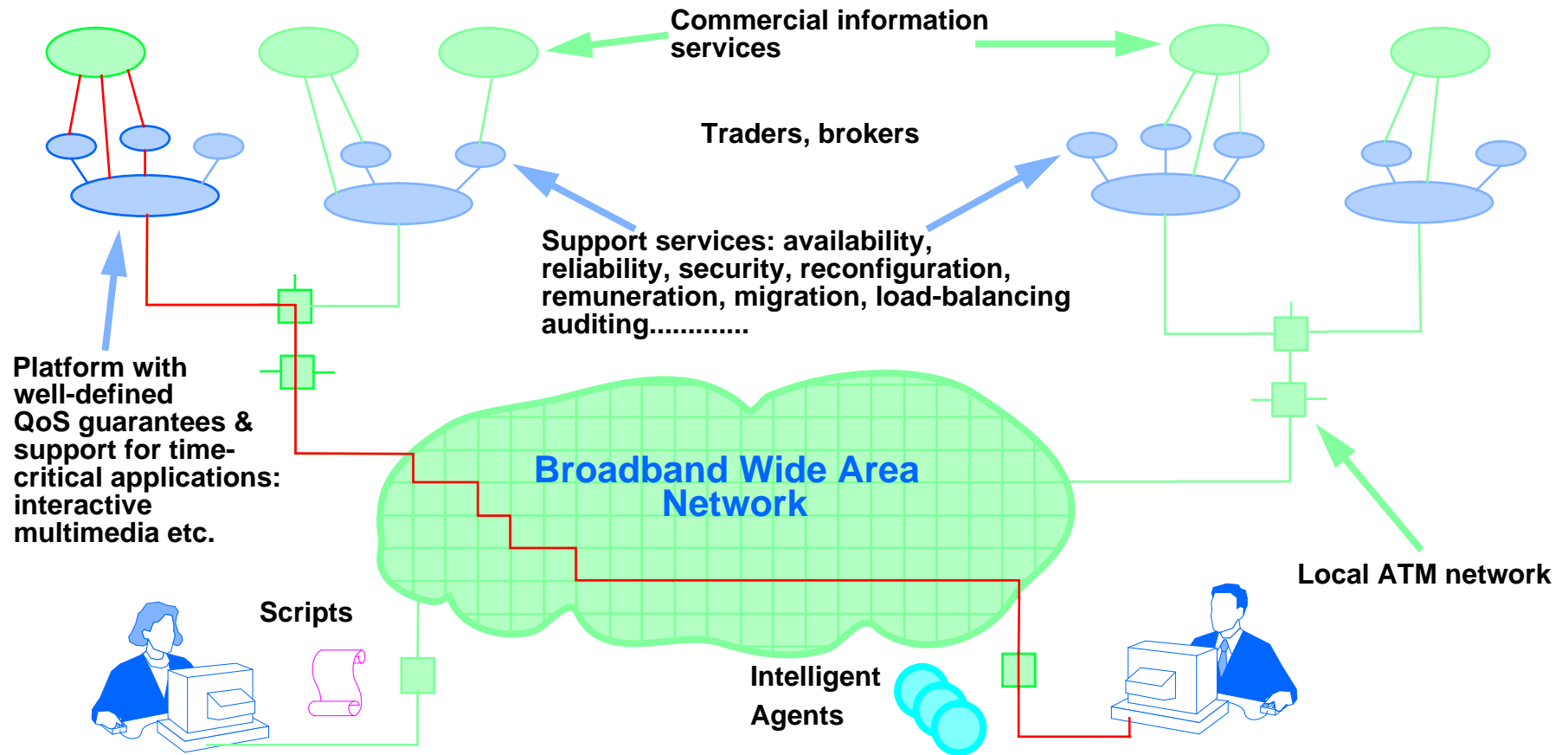


## New Requirements

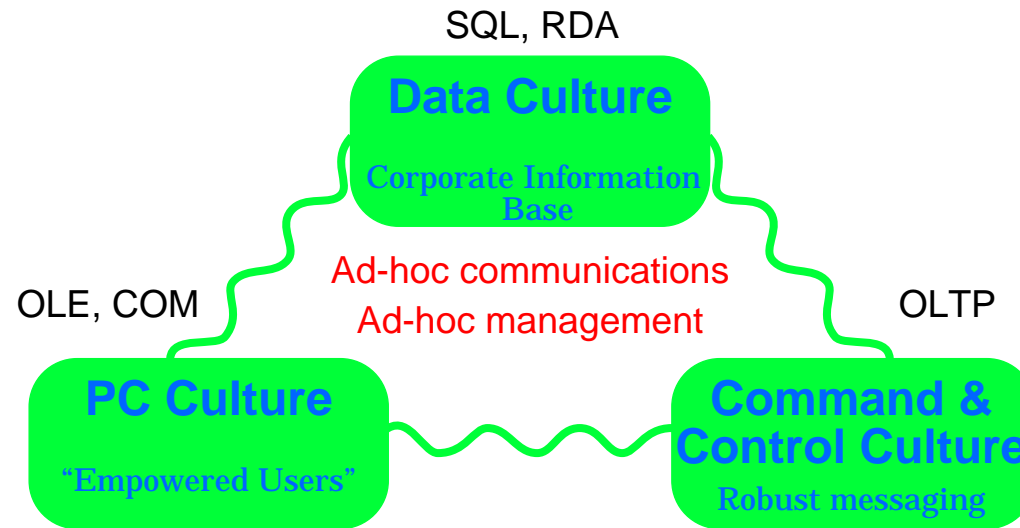




# The Technology



## Distributed Computing Today



- Three important and quite separate distributed computing cultures exist
- Each has its own means of distribution and application integration
- They meet different needs - no single one can absorb the others



## Strengths, Weaknesses

- **Data Culture**

- +/- Remote data access - mostly proprietary remote login SQL access
- +/- Federated databases - typically read-only
- + Stored procedures - for high throughput critical business transactions
- + Object repositories - for more flexible structuring, fine grained locking

- **PC Culture**

- +/- Document sharing (OLE) - scaling issues, passive object model
- +/- Network Administration - scaling and security issues
- +/- Workflow - proprietary, not very robust
- +/- Multi-media - ad hoc, unintegrated

- **Control Culture**

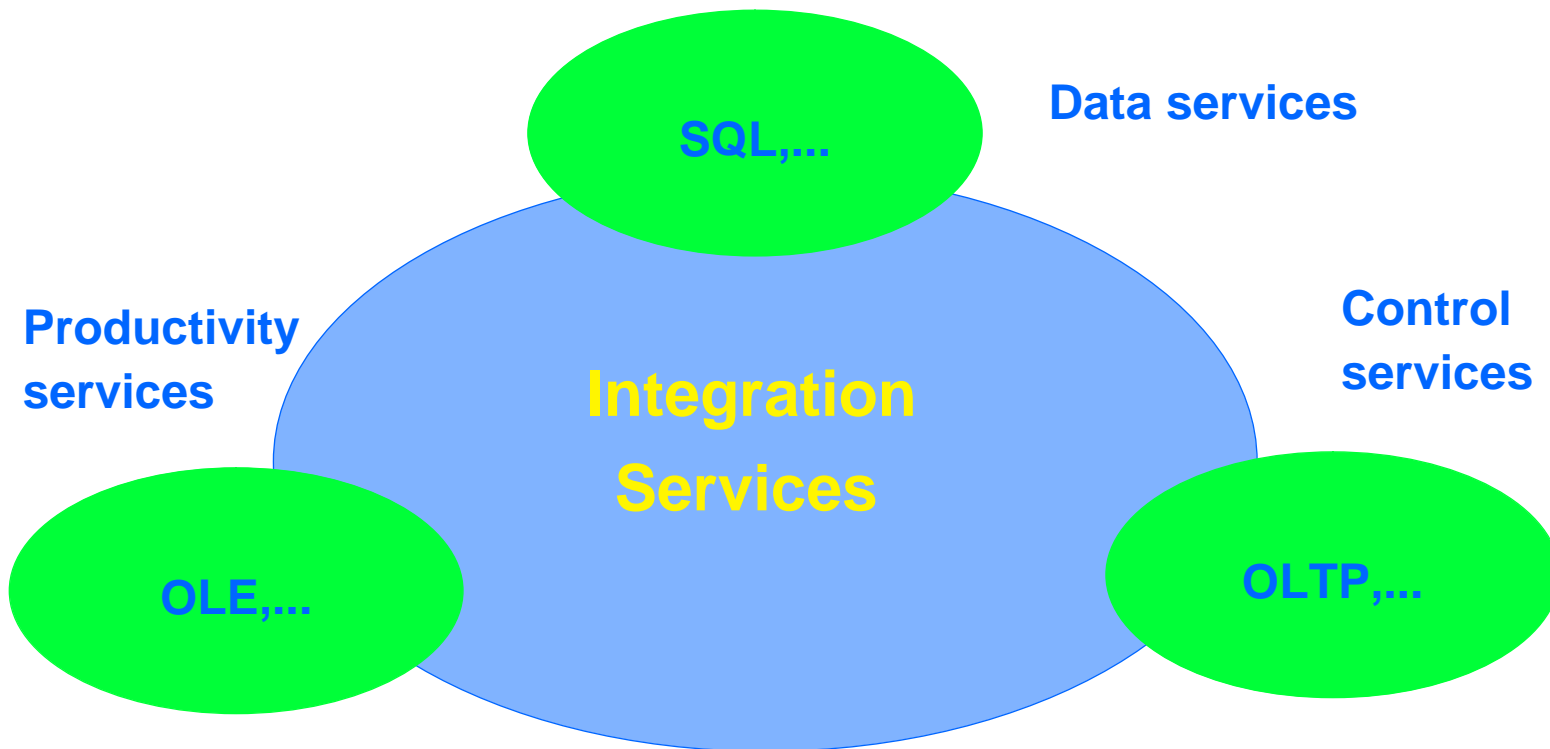
- + On-line transaction processing - reliable message routing and server activation
- + Strong on throughput, failover and security
- Systems programmer oriented API
- +/- Workflow - preconceived application models



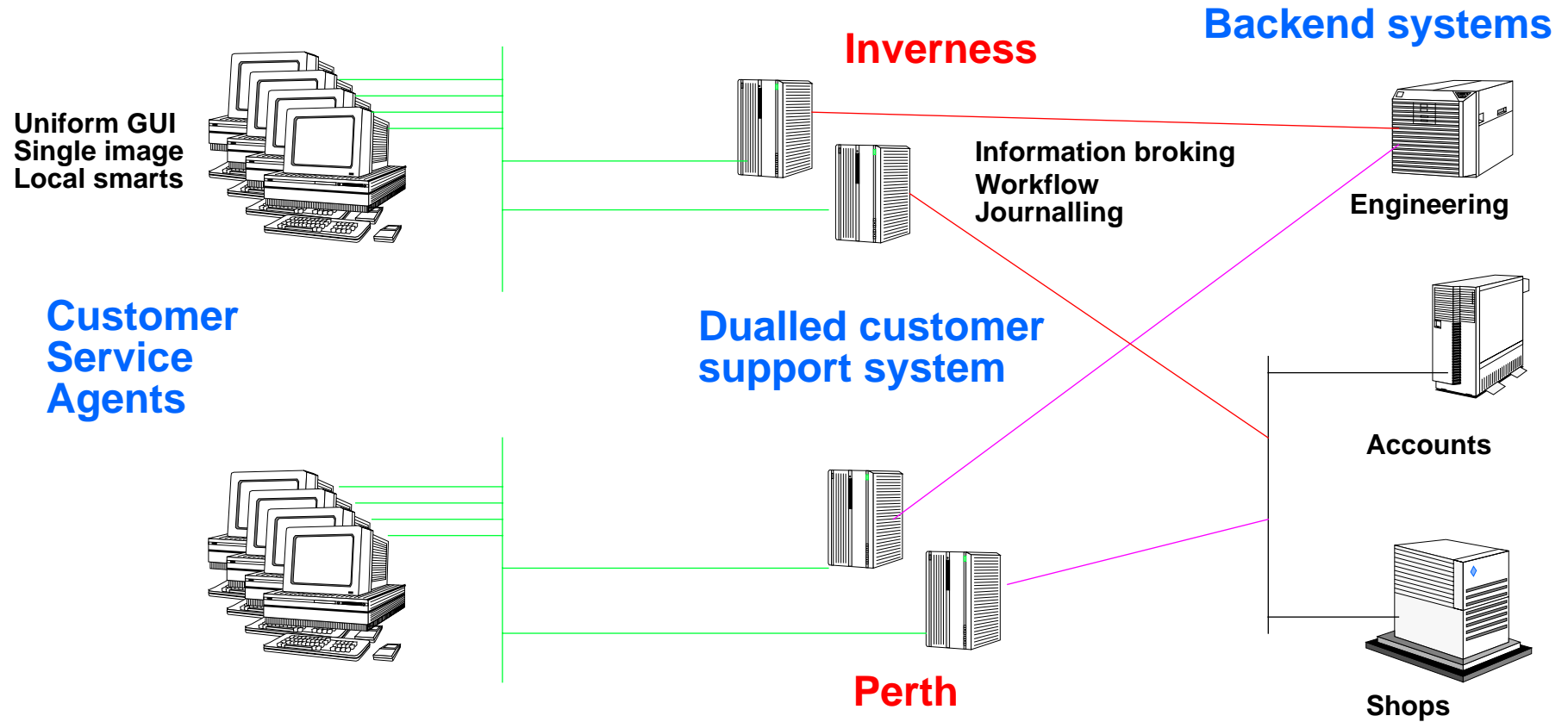
## The Systems Integrator's Wish List

- 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 should only affect performance, 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 - be manageable

# Distributed Computing = Integration Services = Interoperability and Management



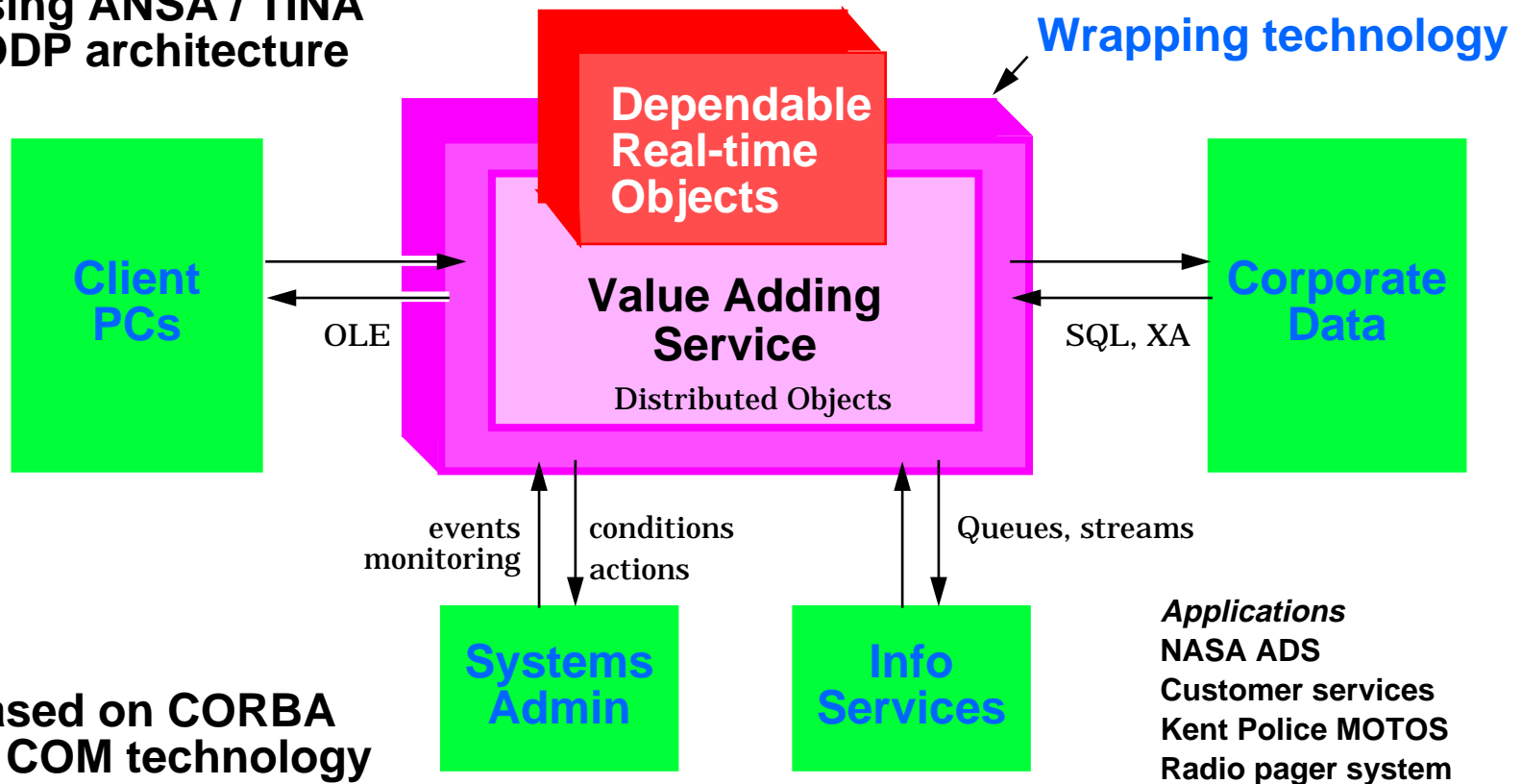
# An Example - Scottish HYDRO





# A Template for Distributed Object Systems

Using ANSA / TINA / ODP architecture



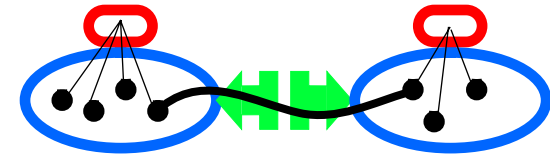
Based on CORBA or COM technology



# Distributed Object Architectures

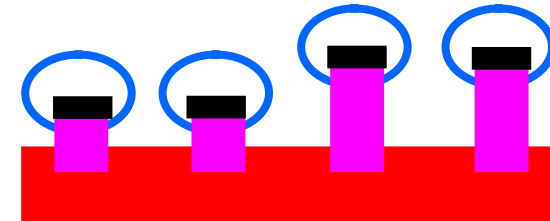
Trading and Federation

Configurable interoperability



Custom Infrastructure

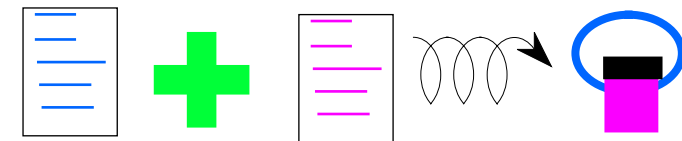
One size does not fit all



Abstract & Automate

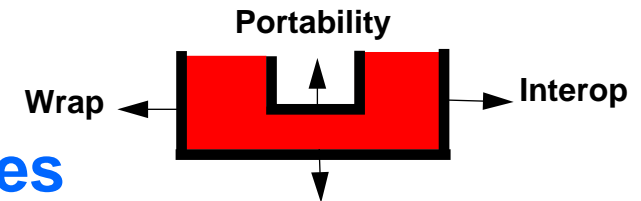
Tools replace APIs

Service Infrastructure



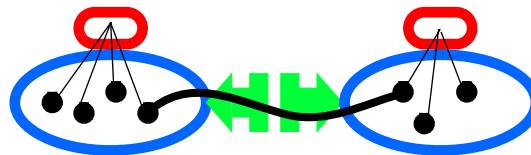
Modular Engineering

Architected internal interfaces





# Trading and Federation





## Domains

- **Large systems are made up of autonomous islands interconnected incrementally**
  - no central authority
  - legacy of old technology
  - conflicting choices of new technology
- **Administrative boundaries**
  - where checks and accounting occur
- **Technology boundaries**
  - where protocol conversion and data translation occur
- **Objects provides services to one another, within domains and between domains**
- **Set up *interceptors* (gateways / bridges) on demand, when trading**



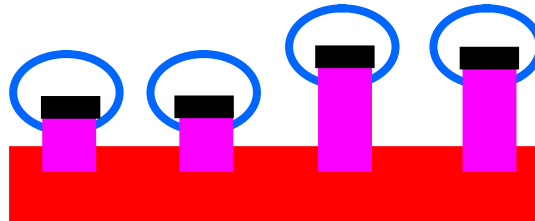
---

## Trading, Binding and Type Safety

- **Trading** - discovering an interface that provides a service
  - *exporters* make service offers, *importers* make service requests
  - *traders* marry imports to exports
  - trading allows new services to be introduced and old services upgraded
- **Binding** - establishing resources for interaction to occur
  - typically implicit for RPC, explicit for streams, groups and where QoS is controlled
- **Type safety** - match signatures to ensure valid interaction
  - “no surprises” rule enables federation
  - done when trading, enforced by compiler



# Custom Infrastructure





## Trade-offs in Distributed Systems

- **Distributed systems engineering is all about trade-offs**
  - **ABSTRACTION** versus **SPECIALIZATION** -the more you hide, the less control you have
  - **CONSISTENCY** versus **AVAILABILITY** - availability implies 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 parts and an open framework into which they slot**
- **Moreover the framework must accomodate coexistence of alternative parts for the same job**

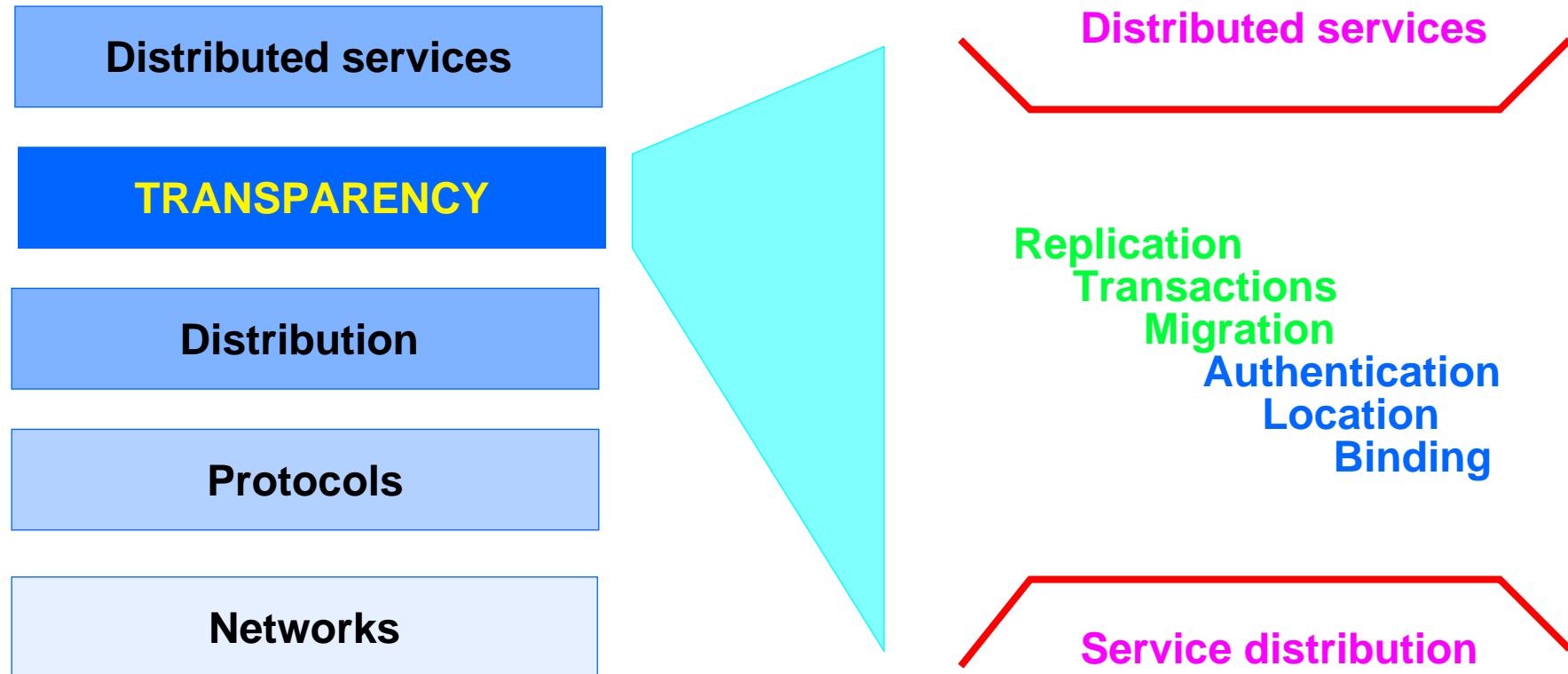


## Selective Transparency

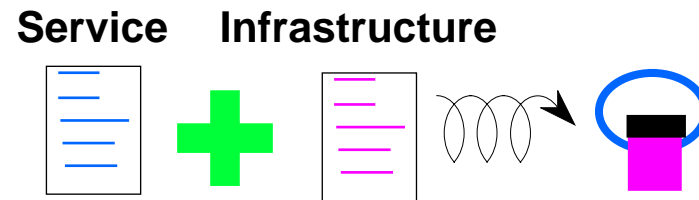
- **Transparency is about hiding irrelevant complexity**
  - **Location** don't need to know where it is to use it
  - **Access** don't need to know how it works to use it
  - **Migration** it can move while you're using it to balance loads or reduce latency
  - **Replication** there may be copies for reliability and/or availability
  - **Persistence** it only gets resources when it needs them
  - **Partial Failure** it always gets to a consistent state
  - **Federation** you don't have to have the same administrator to use it
- **The transparency layer supports the functionality of the basic service distribution layer, and adds additional guarantees**
  - same API for issuing requests
  - extra management functions for controlling transparency



# Engineering Framework



# Abstract and Automate







---

## Distributed Programming is Different

### TRADITIONAL

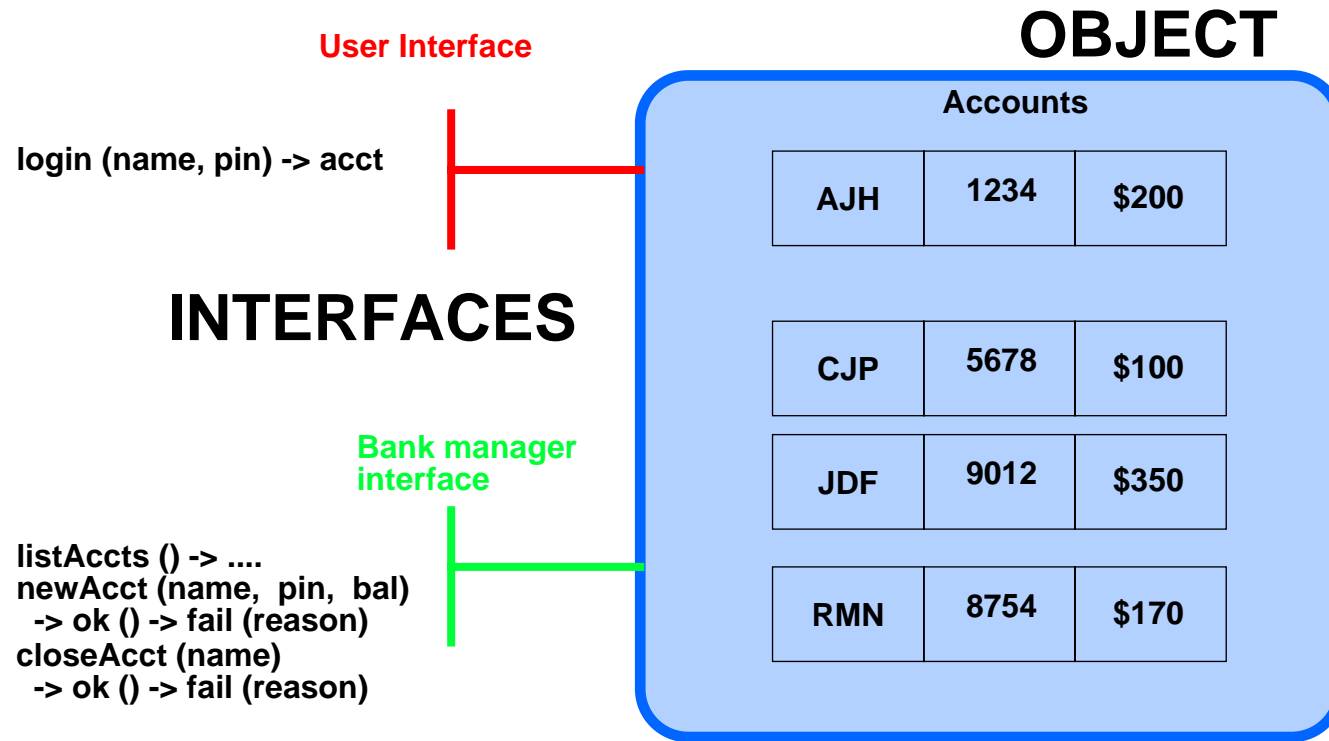
Local  
Sequential  
Single Environment  
Fixed Location  
Single Copy  
Synchronous  
Direct  
Shared  
Global  
Complete failures  
Early Binding

### REVERSED

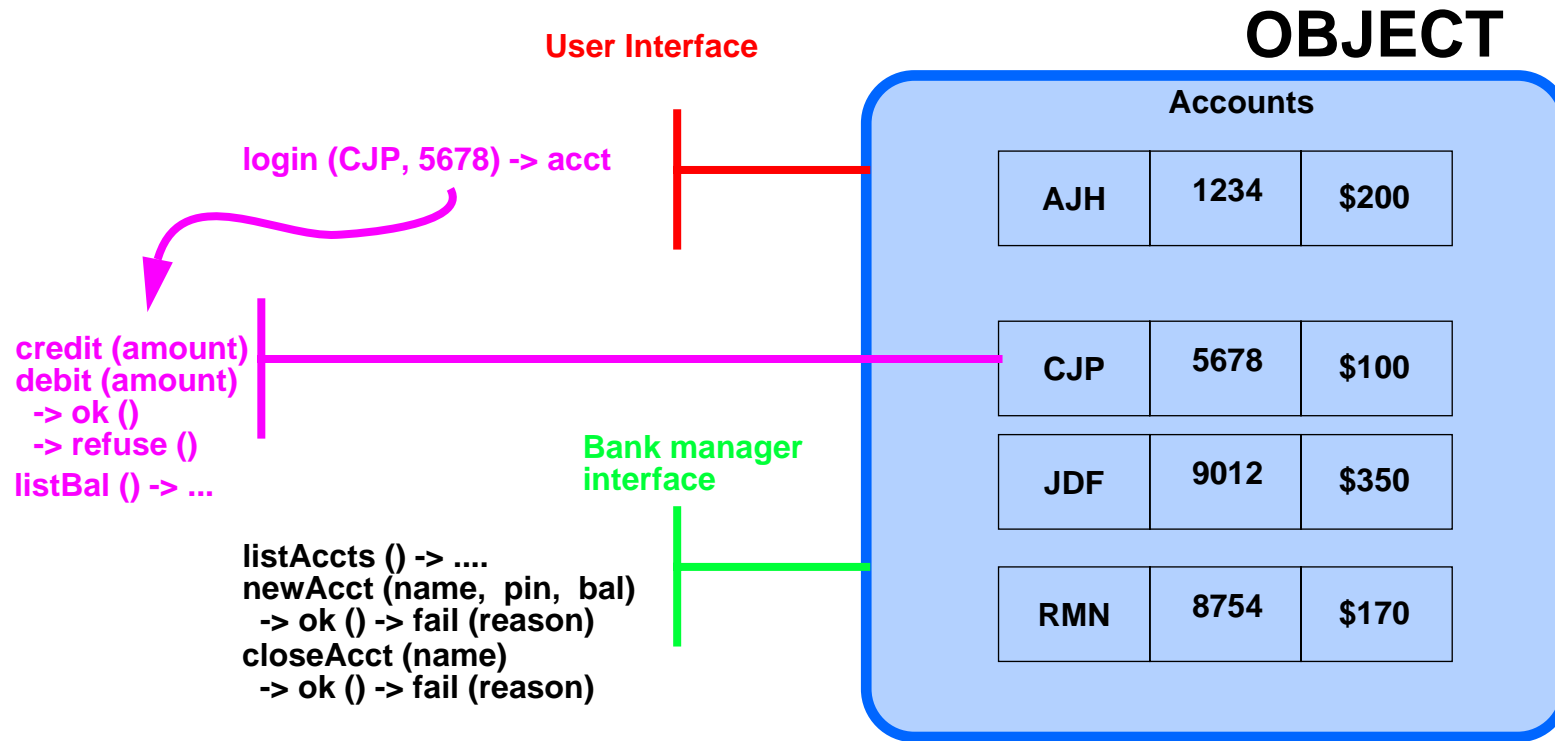
Remote  
Concurrent  
Diverse Environment  
Mobile  
Multiple Copies  
Asynchronous  
Indirect  
Separate  
Context Relative  
Partial Failures  
Late Binding

- Trying to wedge this into a traditional view won't work

# Distributed Object Model (1)



## Distributed Object Model (2)





## Why Do APIs Grow So Large?

- **Rich set of concepts needed for distributed programming**
  - threads
  - requests
  - replication
  - atomicity
  - .....
- **Optimized engineering for common cases**
  - e.g. forked call -> asynchronous call to save a local thread
- **Special engineering for special cases**
  - e.g. spawned atomic call -> start new top level transaction
- **Combinatorial explosion in functions overwhelms the programmer**

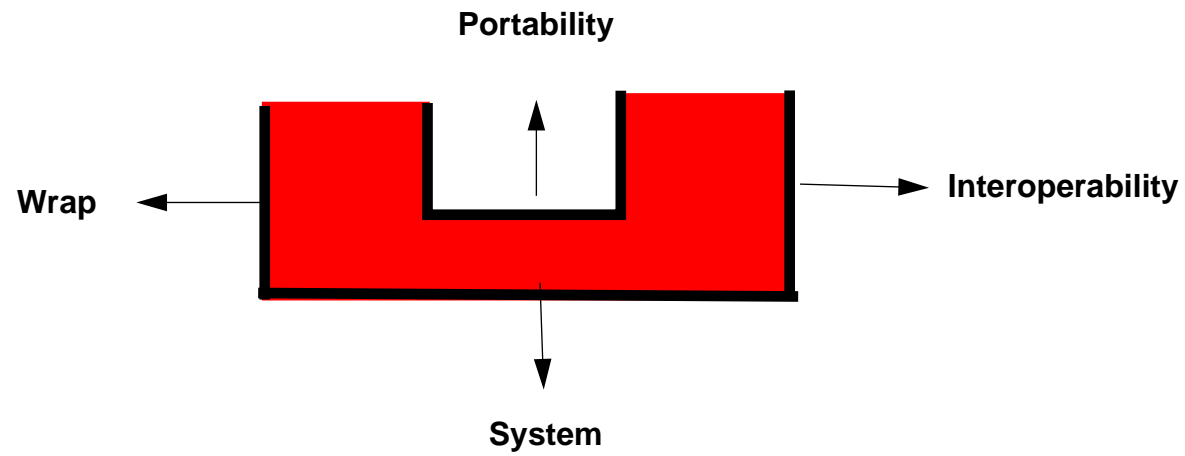


---

## Let Compilers and Tools Take The Strain

- **Exploit abstraction, program in application oriented concepts**
  - **most aspects of OO really help, some hinder**
- **Simple (pre-processor) extensions go a long way**
  - **especially if leveraging an OO language**
- **orthogonality - e.g. “dot” and “bar” vs. threads and RPC API**
  - **languages minimize complexity without losing scope for optimization**
- **declarative - state requirements and policies not mechanisms**
  - **point already proven by IDLs and stub generators**
  - **decouple applications from engineering - ANSA PREPC experience**
- **strong type checking for safety and confidence**

# Modular Engineering





## The Rest of the Picture

- So far we have covered technology issues
- The complete ANSA Architecture has five viewpoints of which computation and engineering are only two
  - the other viewpoints bring in business models and requirements
  - the other viewpoints emphasize the need for tools and online system meta data
- The broader framework has been carried into the ISO/ITU Reference Model for Open Distributed Processing

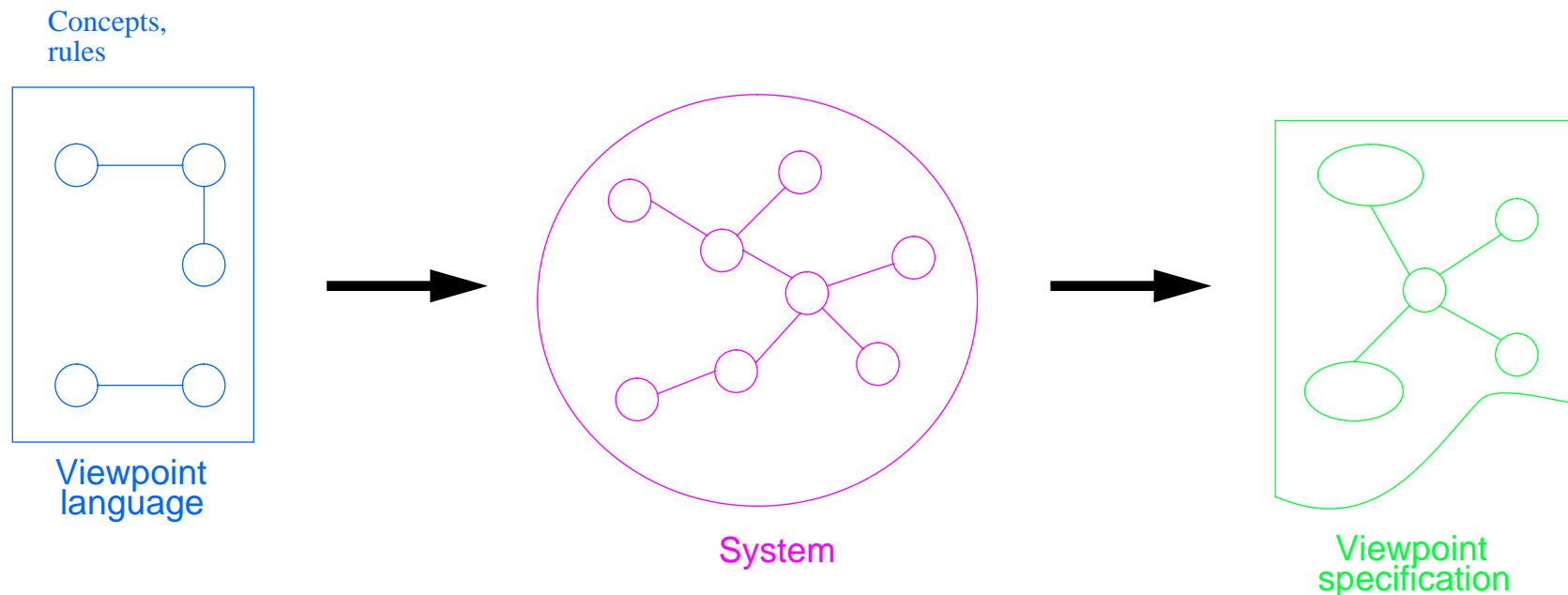


## Languages and Viewpoints (1)

- **ODP Framework** consists of five **viewpoint languages** and a set of **infrastructure functions**
- Each viewpoint language enables specification of an ODP system or component
- Languages are structured so that **consistency checking** between alternative viewpoint specifications is possible
- The chosen set is **necessary** and **sufficient** for needs of ODP
  - clear separation of concerns (requirements, design, implementation)
  - maximum scope for federation, transparency and modularity
- Each language consists of **concepts** (vocabulary) and **rules** (grammar)



## Languages and Viewpoints (2)



- **Mathematical basis in *projection* of a set of concepts and abstraction/specialisation relations over a semantic net.**



## Viewpoints

- **Enterprise** - purpose, scope and policies for a system
  - business process models, policy models
- **Information** - kinds on information handled by the system and constraints on the use and interpretation of that information
  - data models, data flows, data consistency constraints
- **Computational** - functional decomposition into objects suitable for distribution
  - functional design
- **Engineering** - the infrastructure required to support distribution
  - implementation block diagram
- **Technology** - the choice of technology to support distributio
- **Don't equate viewpoints to design process refinement steps!**



---

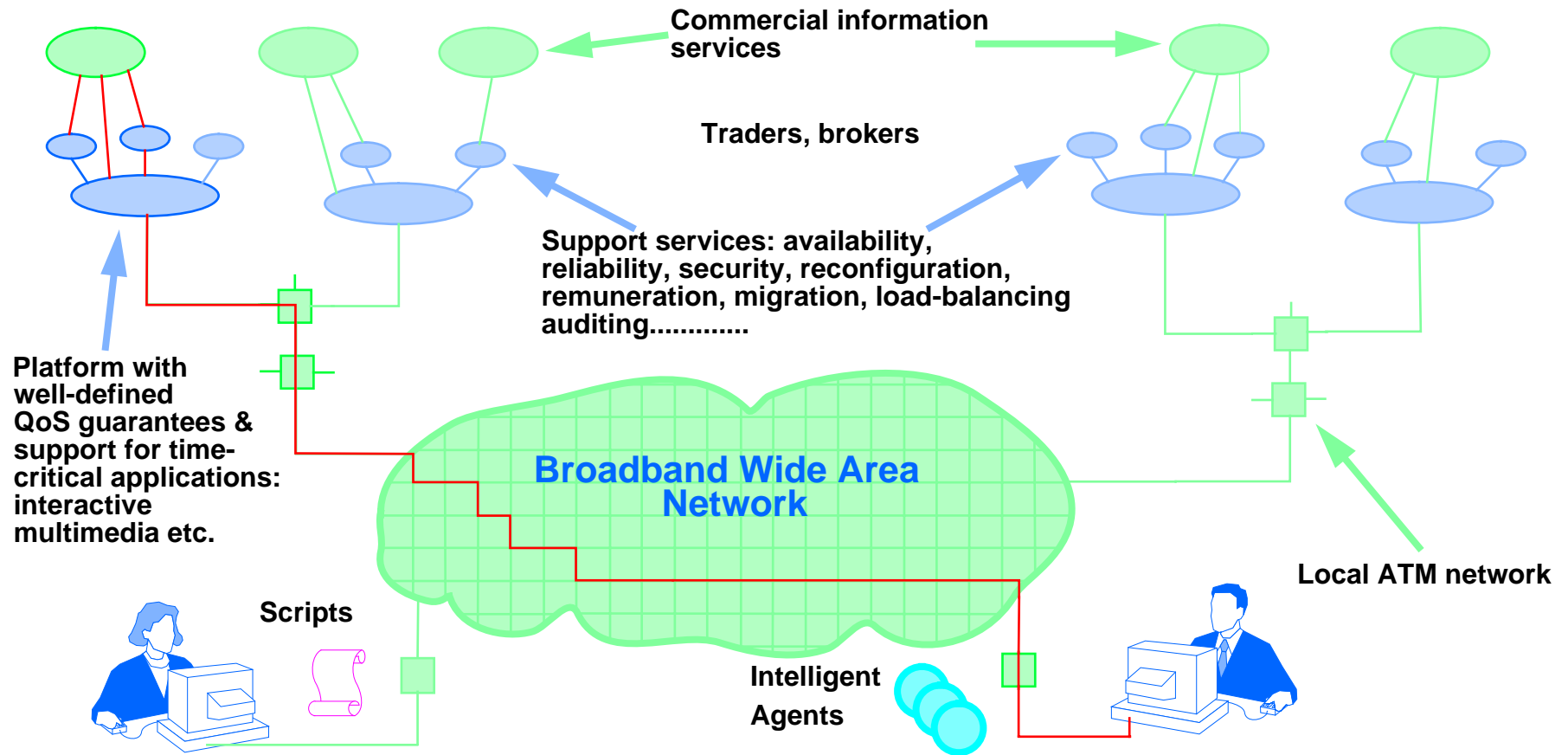
## Is ANSA Real?

- All except security has been built for the ANSAware prototype
- ANSAware anticipated DCE RPC, IDL and threads by 2 years
- ANSAware anticipated CORBA, object life cycle and UNO by 2 years
- ANSAware has fed into products by HP, ICL, BT, Bellcore, GPT, AEG
- ANSAware '93 included DCE RPC and threads coexisting with previous home-grown solutions
- ANSAware '94 development majored on real-time and streams
- ANSAware '95 development will major on WWW interceptors and repository



# Summary

# The Vision





## How APM can help

- **ANSA architecture for Open Systems**
  - trading and federation
  - selective transparency
  - abstract and automate
  - modular engineering
  - controlled interoperability
  - one size does not fit all
  - tools replace APIs
  - architected internal interfaces
- **developed by the ANSA consortium**
  - proved in real products and systems
  - consistent with key industry standards
  - with a firm grip on the future
- **APM Services**
  - ANSA research, ANSA “Object Laboratory”
  - ANSA Consulting, ANSA training, ANSAware