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

Distributed Computing meets the Information Superhighway

Andrew Herbert

Abstract

This a presentation for a talk to be given at the ICL / University of Newcastle workshop on “The Future of Software”, September 5-8th 1995.

IIt explores the ways in which object technology can improve both the engineering and the capabilities of technologies like the World Wide Web.

APM.1556.03 **Approved** 25th August 1995
External Paper

Distribution:
Supersedes:
Superseded by:

Distributed Computing meets the Information Superhighway

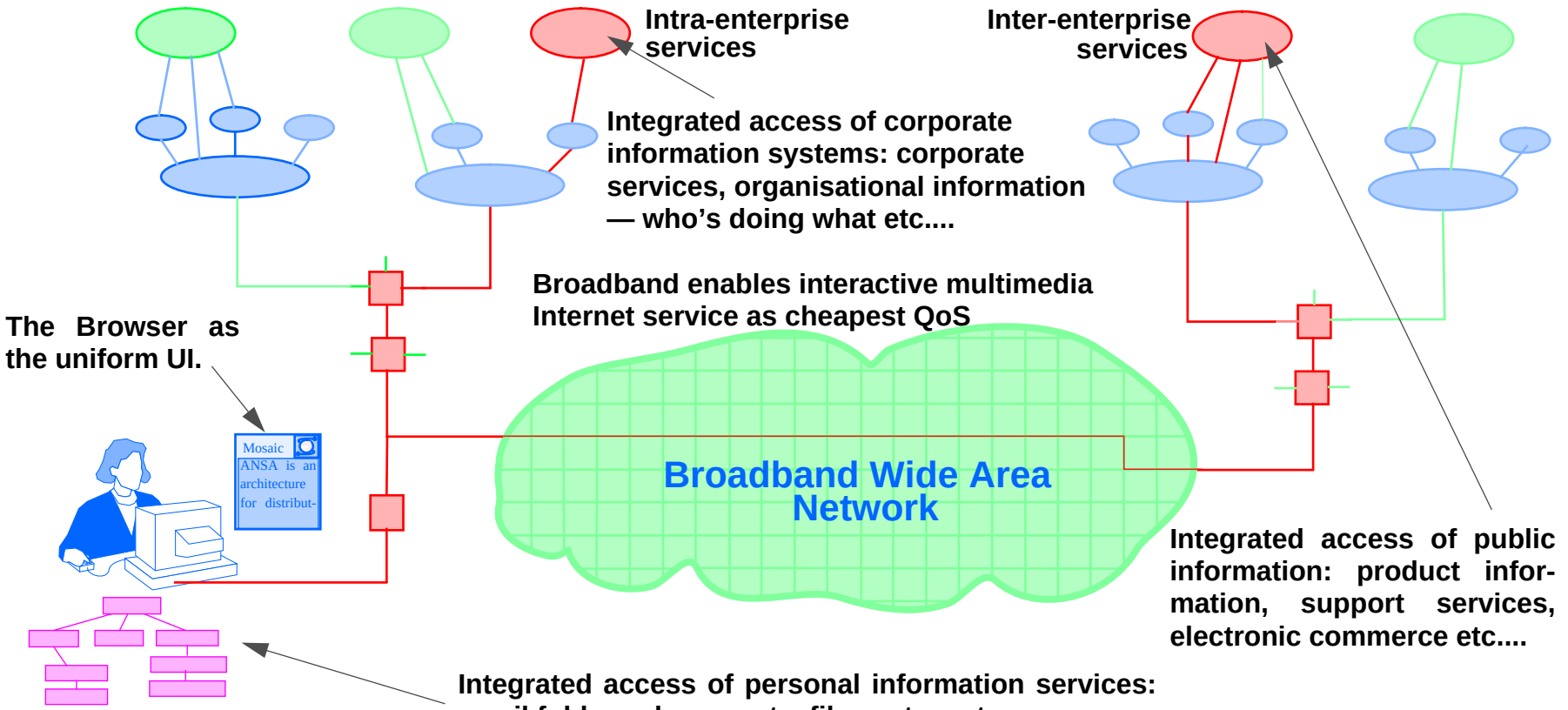
Andrew Herbert

ANSA / APM

From Global Network to Global Operating System

- **Observation:**
  - The World Wide Web is creating a “Uniform” information space and a platform for electronic commerce
- **Proposition 1:**
  - Distributed systems technology can overcome engineering weaknesses in the current Web
- **Proposition 2:**
  - Distributed systems technology can extend the capabilities of the current Web
- **The next frontier:**
  - Active content and semantics-driven information processing

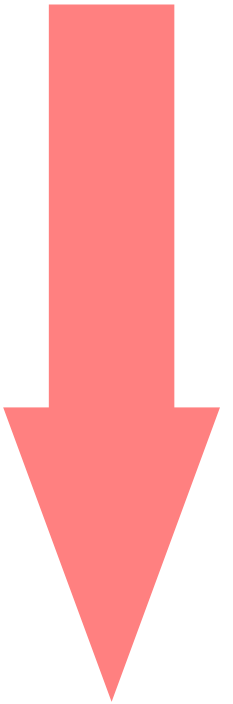
Creating a uniform information space



Technology requirements for the uniform information space

Well understood

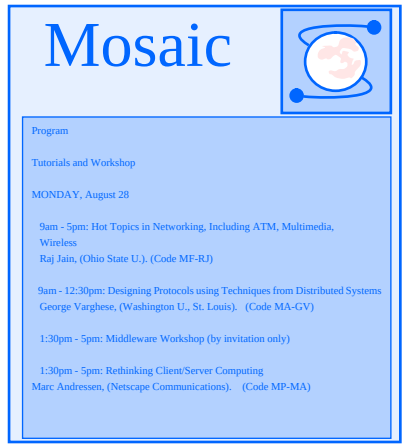
- Presentation (= Browsers + HTML, postscript etc.)
- Creation (= Authoring tools)
- Efficient protocols, interactive multimedia
- Extensible front ends (making new functionality available)
- Extensible back ends (transactional services)
- Dependability (availability, security, integrity)
- Navigation tools (finding the information)
- Administration tools (managing the services)



Poorly understood

An example “transactional” application

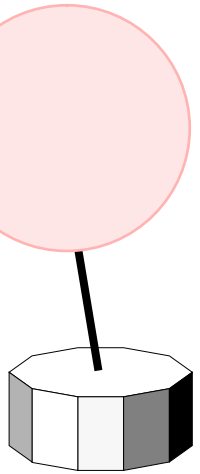
- **Scenario: booking theatre tickets for multiple events.**
- **Requirements**
  - **Browsing program information**
  - **Choice of seats & dates**
  - **On-line selection of preferences**
  - **Confirmation of booking by server**
  - **Ability to change booking**
  - **Single payment on confirmation of booking by client**



User interface: a web browser (e.g. Mosaic)

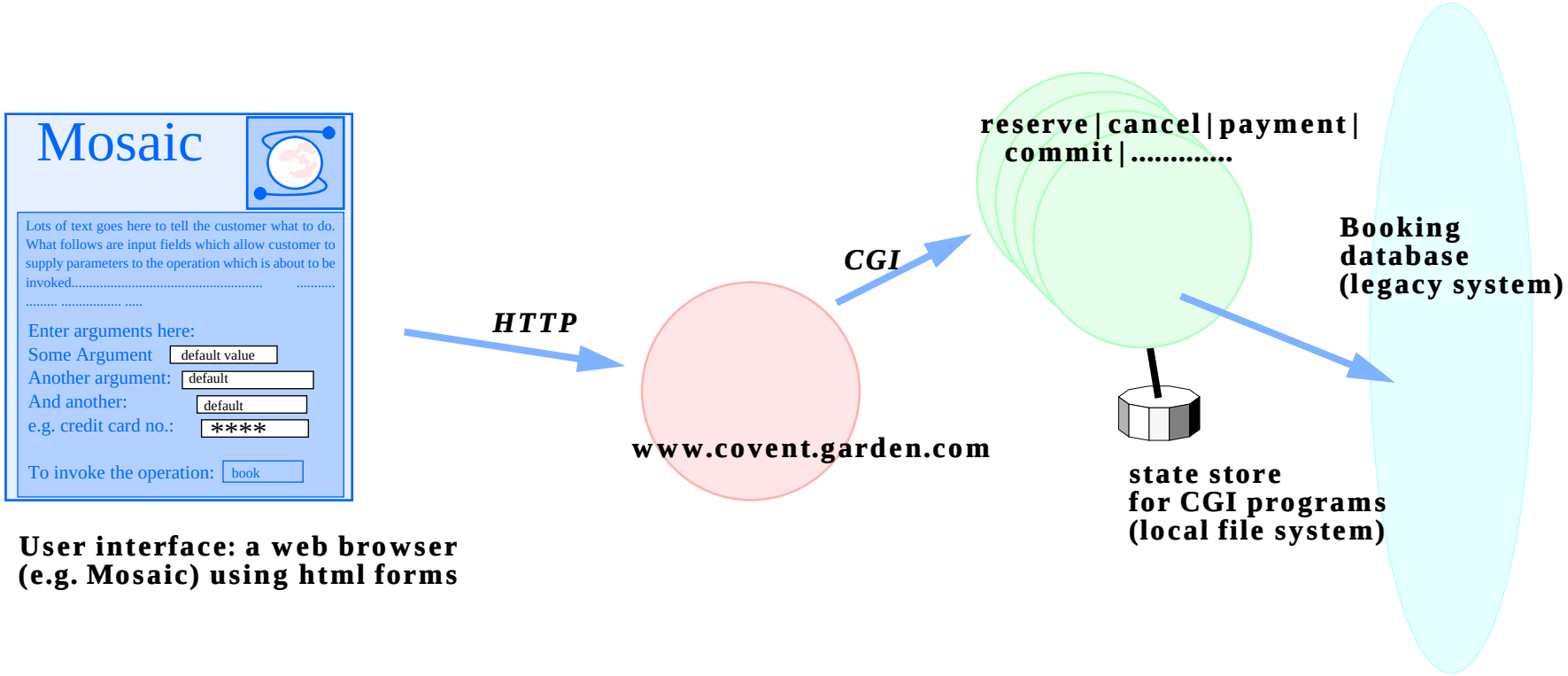
www.covent.garden.com

HTTP



Document store containing program information

How it's done today.....

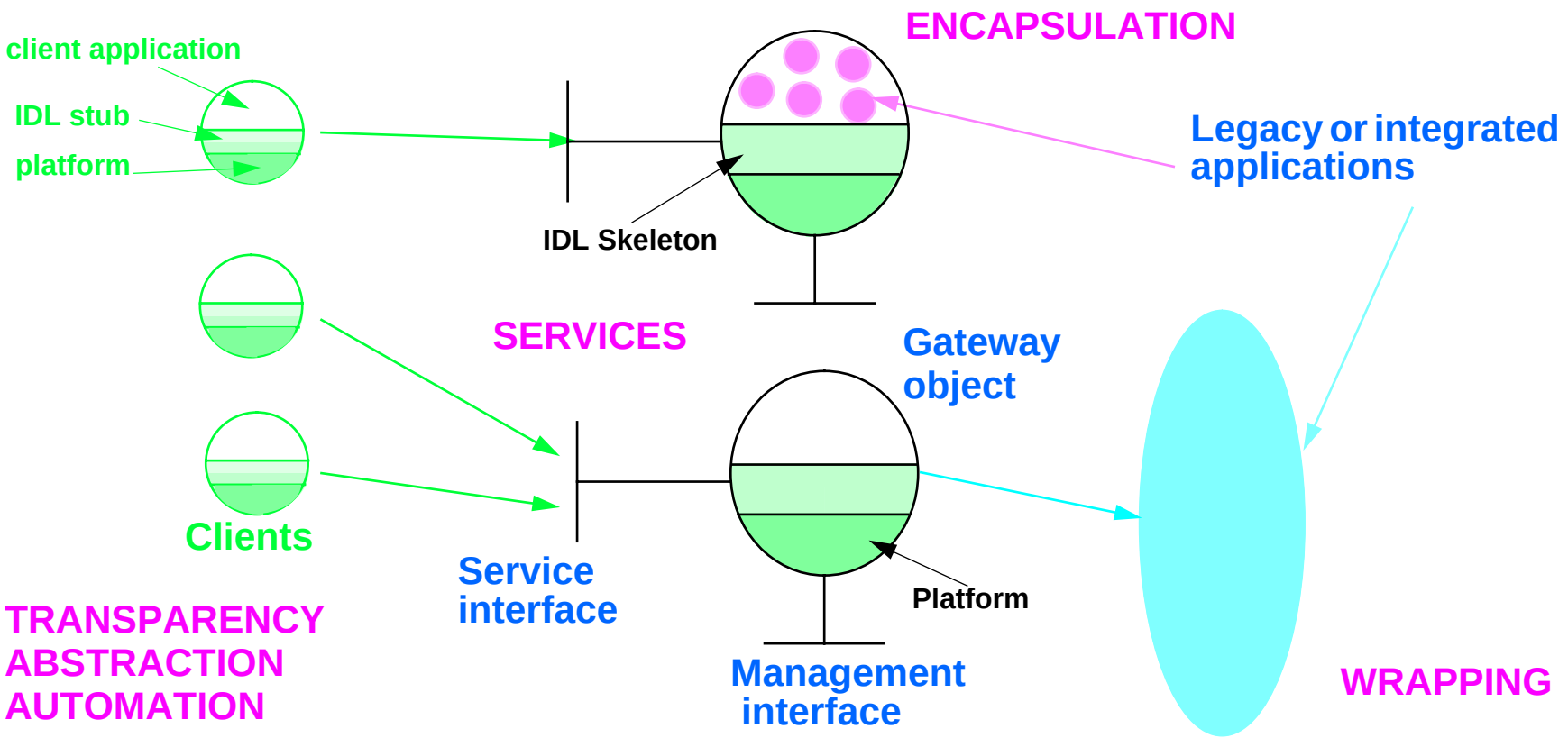


User interface: a web browser (e.g. Mosaic) using html forms

A Hacker's Paradise

- **The CGI allows HTTP servers to communicate with other programs**
  - **It is hard (error prone) to write the code to unmarshal the parameters to the CGI programs (no tool support)**
  - **CGI Program is "forked" for each interaction, no "session" structure**
- **CGI driven by HTML forms**
  - **Care is needed to make sure the HTML form and corresponding CGI program are consistent (both in numbers and "types" of parameters).**
  - **For most browser, customisation of the user interface is limited by what can be displayed in browsers using HTML forms technology — agent technology (e.g. Java) allows greater customisation (see demonstration).**

The Solution: Distributed Object Technology



Benefits of Object Technology

- **Access services through defined (in IDL) interfaces**
  - stub compilers can generate IDL stubs and skeletons which abstracts the programmer from the underlying protocols and API.
  - calling a remote service a procedure call; the underlying protocols and the internals of the platform are hidden
- **We can slide in infrastructure services transparently**
  - transactions, authentication, replication, migration
- **We can apply object management functions**
  - everything is an object
  - life cycle, event handling, repository, property, query, trading
- **We can substitute alternative protocols and data formats**

ANSWeb Phase 1 - A Demonstrator

Browser

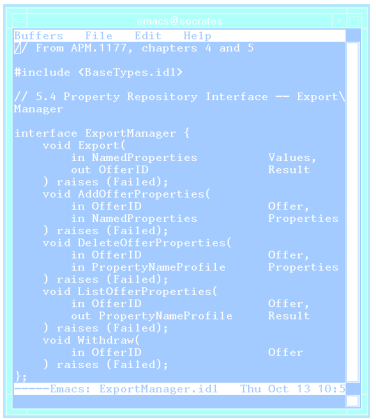
HTTP Server

Application

IDL Skeleton

CGI program

CORBA IDL for Service



Generate
Form (on the fly)

Generate
IDL stubs for clients

Generate IDL
skeletons for servers

IDL stub
libwww

Bespoke client

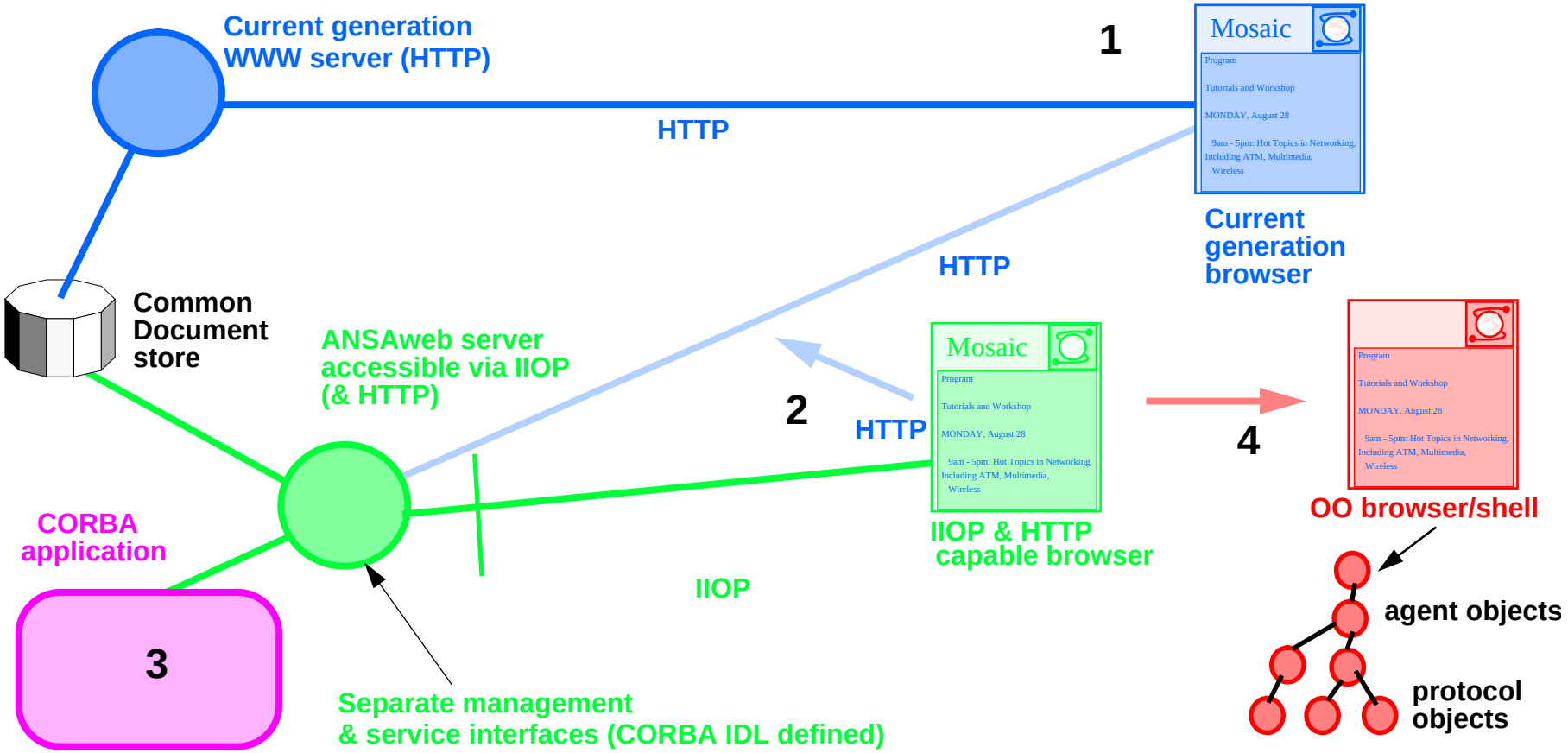
E.G. HTML Form Generation

```
interface Echo{
   string Echo(in string Src);
   void Sink(in string Src);
   string Source(in long Length);
   string Reverse(in string Src);
};
```

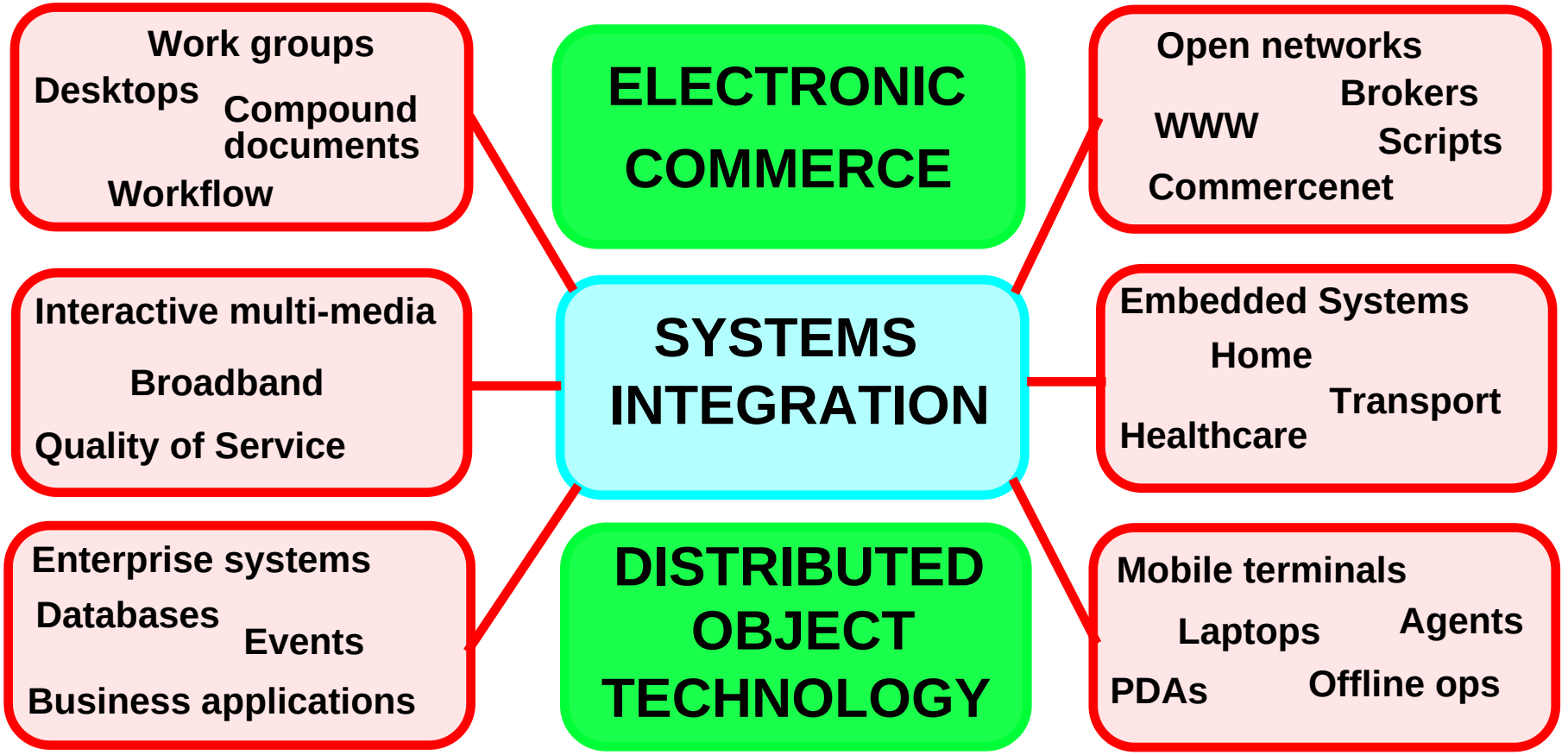
Stub Compiler generates this HTML form

```
<head>
<TITLE>Input for Echo</TITLE>
</head>
<BODY><H1>Input for Echo</H1>
<HR>

<H2> Operation Echo</H2>
<FORM METHOD=”POST” ACTION=”http://socrates.ansa.co.uk:8080/cgi-bin/Echo”>
<P>Enter arguments here:<P>
<INPUT NAME=”Operation” TYPE=hidden VALUE=”Echo_Echo”>
CORBA_string Src: <INPUT SIZE=10 NAME=”Src”> <P>
To invoke Echo_Echo: <INPUT TYPE=”submit” VALUE=”Echo_Echo”><P>
</FORM>
<HR>

<H2> Operation Sink</H2>
<FORM METHOD=”POST” ACTION=”http://socrates.ansa.co.uk:8080/cgi-bin/Echo”>
<P>Enter arguments here:<P>
<INPUT NAME=”Operation” TYPE=hidden VALUE=”Echo_Sink”>
CORBA_string Src: <INPUT SIZE=10 NAME=”Src”> <P>
To invoke Echo_Sink: <INPUT TYPE=”submit” VALUE=”Echo_Sink”><P>
</FORM>
<HR>

<H2> Operation Source</H2>
<FORM METHOD=”POST” ACTION=”http://socrates.ansa.co.uk:8080/cgi-bin/Echo”>
<P>Enter arguments here:<P>
<INPUT NAME=”Operation” TYPE=hidden VALUE=”Echo_Source”>
CORBA_long Length: <INPUT SIZE=10 NAME=”Length”> <P>
To invoke Echo_Source: <INPUT TYPE=”submit” VALUE=”Echo_Source”><P>
</FORM>
<HR>

<H2> Operation Reverse</H2>
<FORM METHOD=”POST” ACTION=”http://socrates.ansa.co.uk:8080/cgi-bin/Echo”>
<P>Enter arguments here:<P>
<INPUT NAME=”Operation” TYPE=hidden VALUE=”Echo_Reverse”>
CORBA_string Src: <INPUT SIZE=10 NAME=”Src”> <P>
To invoke Echo_Reverse: <INPUT TYPE=”submit” VALUE=”Echo_Reverse”><P>
</FORM>
</BODY>
```

ANSAweb Phase 2: Migrating the web to Distributed Objects



The Future of Software - Systems Integration



New Requirements

Interactive Multi-media

Performance
Streams
Throughput
Resource control
QoS negotiation

Open Networks

Federated naming
Intelligent broking and trading
Cooperative, autonomous management
Security

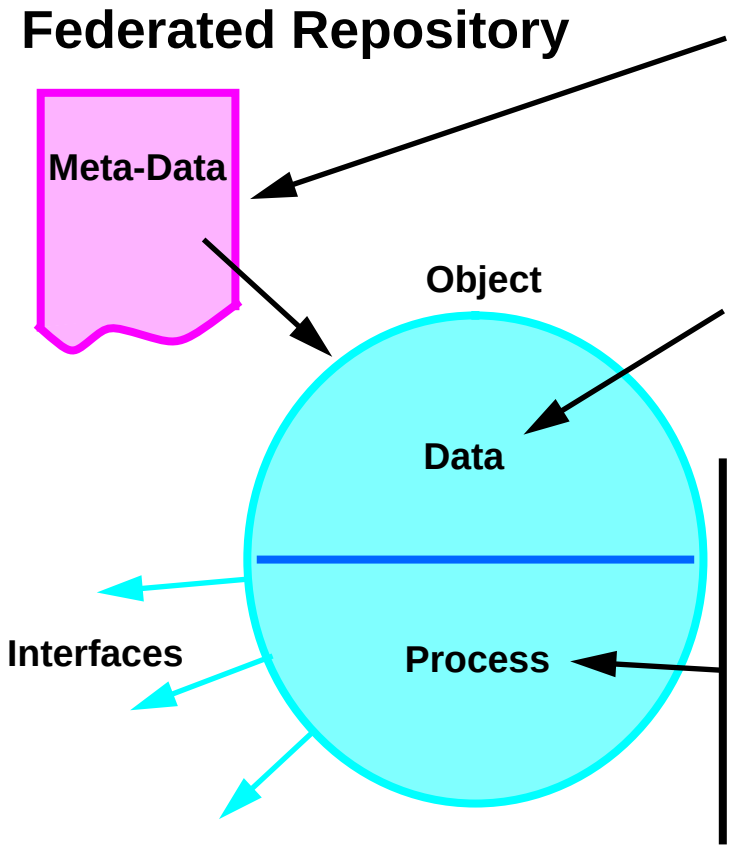
Distributed Information

Intelligent information filters and agents
Business monitoring
Information servers
Computer assisted business processes

Embedded Systems

Down scaling
Event processing
Low power usage
Predictable scheduling

Active Content



- automate management, navigate, filter, monitor

  Semantics-based processing

- the content
  - life > 100 years
  -
- protect integrity of data
  - objects protect themselves
  - objects manage thenselves
  - replace / upgrade ‘in service’
- choose best presentation

Conclusions

- Distributed object interconnect
- Focus on systems integration
- Small distributed systems within vast networks
- Federations of autonomous systems rather than a global distributed operating system
- Interactive multimedia bring QoS management to the Internet
- Active content is the applications paradigm
- The contenders: OMG with CORBA, MicroSoft with OLE 2.0 & (D)COM