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.

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

The Browser as the uniform UI.

Broadband enables interactive multimedia Internet service as cheapest QoS

Integrated access of corporate information systems: corporate services, organisational information — who’s doing what etc....

Integrated access of public information: product information, support services, electronic commerce etc....

Integrated access of personal information services: email folders, documents, file system etc....

Broadband Wide Area Network

Intra-enterprise services

Inter-enterprise services

Personal information services

Mosaic: ANSA is an architecture for distribut...
Technology requirements for the uniform information space

- 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)

Well understood

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

Lots of text goes here to tell the customer what to do.
What follows are input fields which allow customer to supply parameters to the operation which is about to be invoked...

Enter arguments here:
Some Argument : default value
Another argument: default
And another: e.g. credit card no.: default
To invoke the operation: book
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

- **Encapsulation**
- **Services**
- **Transparency Abstraction Automation**
- **Wrapping**

Diagram:
- Client application
- IDL stub
- Gateway object
- Service interface
- Management interface
- Legacy or integrated applications
- Platform
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

- Generate IDL skeleton for servers
- Generate IDL stubs for clients
- CGI program
- IDL skeleton
- Application
- HTTP Server
- CORBA IDL for Service
- Generate Form (on the fly)
- Bespoke client
- IDL stub libwww
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

```html
<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 type="text" 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 type="text" 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 type="text" 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 type="text" 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

Current generation WWW server (HTTP)

Common Document store

ANSAweb server accessible via IIOP (& HTTP)

Current generation browser

HTTP

IIOP & HTTP capable browser

OO browser/shell

agent objects

protocol objects

Separate management & service interfaces (CORBA IDL defined)
The Future of Software - Systems Integration

- Work groups
- Desktops
- Compound documents
- Workflow

- Interactive multi-media
- Broadband
- Quality of Service

- Enterprise systems
- Databases
- Events
- Business applications

- Electronic Commerce
- Systems Integration
- Distributed Object Technology

- Open networks
- Brokers
- WWW
- Scripts
- Commercenet

- Embedded Systems
- Home
- Transport
- Healthcare

- Mobile terminals
- Laptops
- Agents
- PDAs
- Offline ops
New Requirements

Interactive Multi-media
- Performance
- Throughput
- Resource control
- QoS negotiation
- Streams

Open Networks
- Federated naming
- Cooperative, autonomous management
- Security
- Intelligent broking
- and trading

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

Embedded Systems
- Down scaling
- Predictable scheduling
- Low power usage
- Event processing
Active Content

Federated Repository

- automate management, navigate, filter, monitor

Semantics-based processing

- the content
  - life > 100 years

- protect integrity of data
  - objects protect themselves
  - objects manage themselves
  - 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