**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**

# The WWW Deliverables

**Nigel Edwards**

**Abstract**

This presentation discusses the WWW deliverables planned for 1995.

The first of these deliverables is the ANSAweb stub compiler (ANSAweb Phase 1). An early version of this was released on June 9th 1995.

Release of ANSAweb Phase 2 is scheduled for December 1995: details of this deliverable are confidential to the consortium.

APM.1517.01 **Approved** 7th July 1995

Project Management (confidential to ANSA consortium for 2 years)

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**Superseded by**:

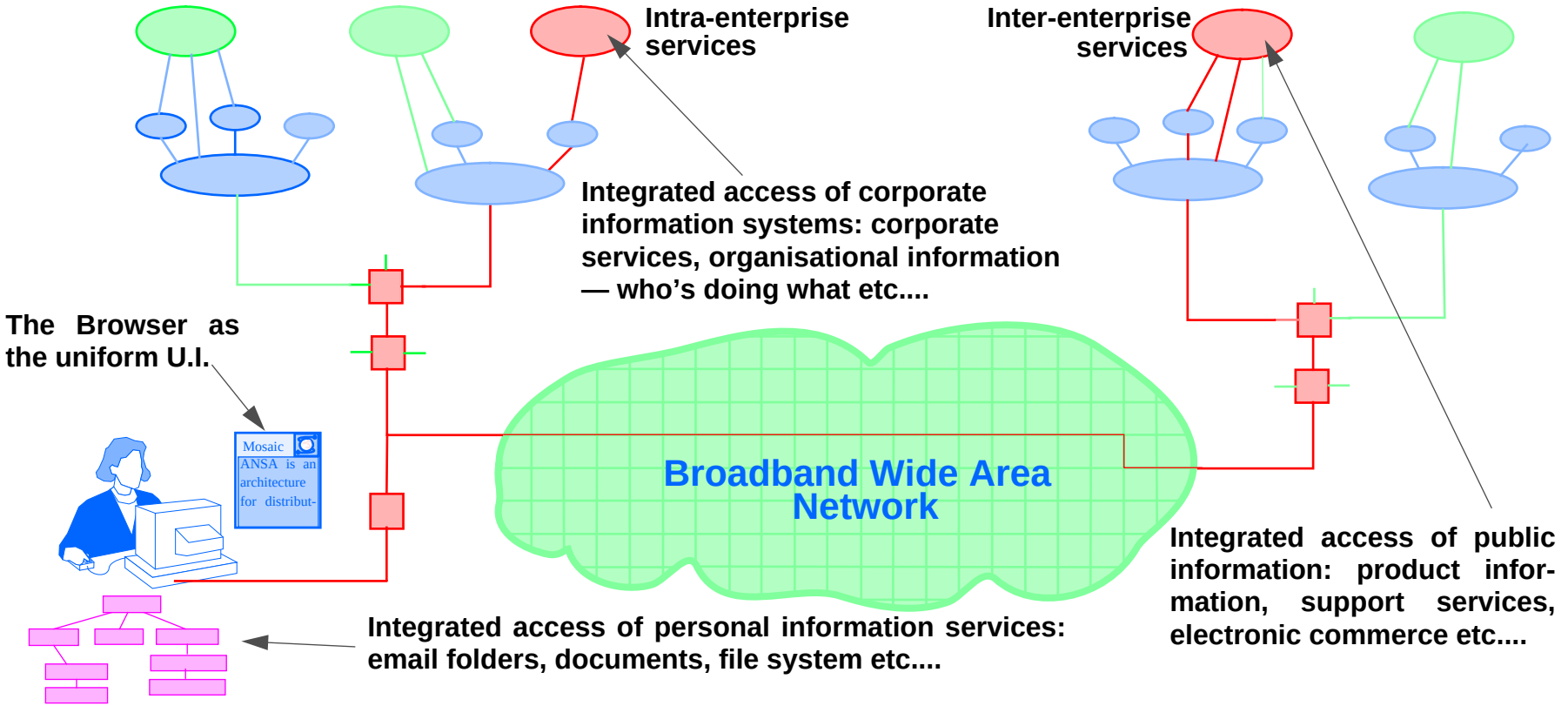

# The WWW Deliverables

Nigel Edwards


# Introduction

- How WWW is creating a “Uniform” information space
- What technology is needed?
- How distributed objects can help
  - What we have done
  - What we are doing

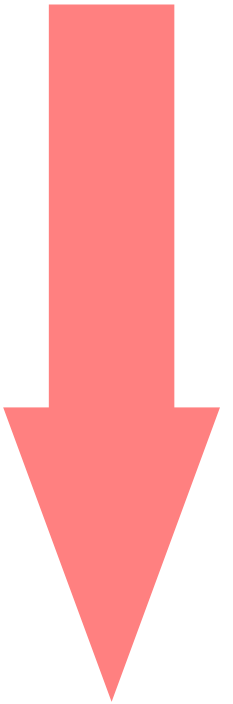
# Creating a uniform information space




# Technology requirements for the uniform information space

Well understood

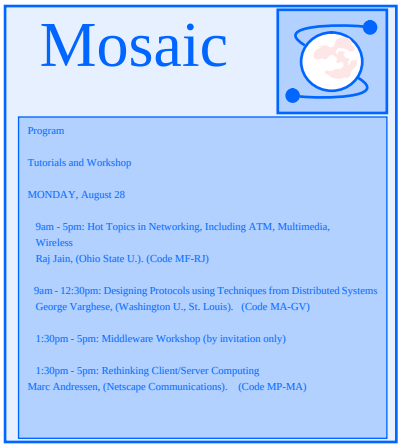
- Presentation (= Browsers + HTML, postscript etc.)
- Creation (= Authoring tools)
- Efficient protocols
- Extensible front ends (making new functionality available)
- Extensible back ends (integrating new & existing systems)
- Navigation tools (finding the information)
- Administration tools (managing the services)



Poorly understood

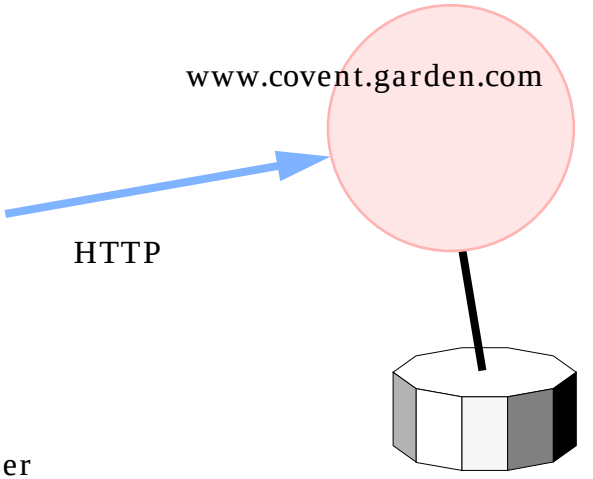
# A “commercial” application

- **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**



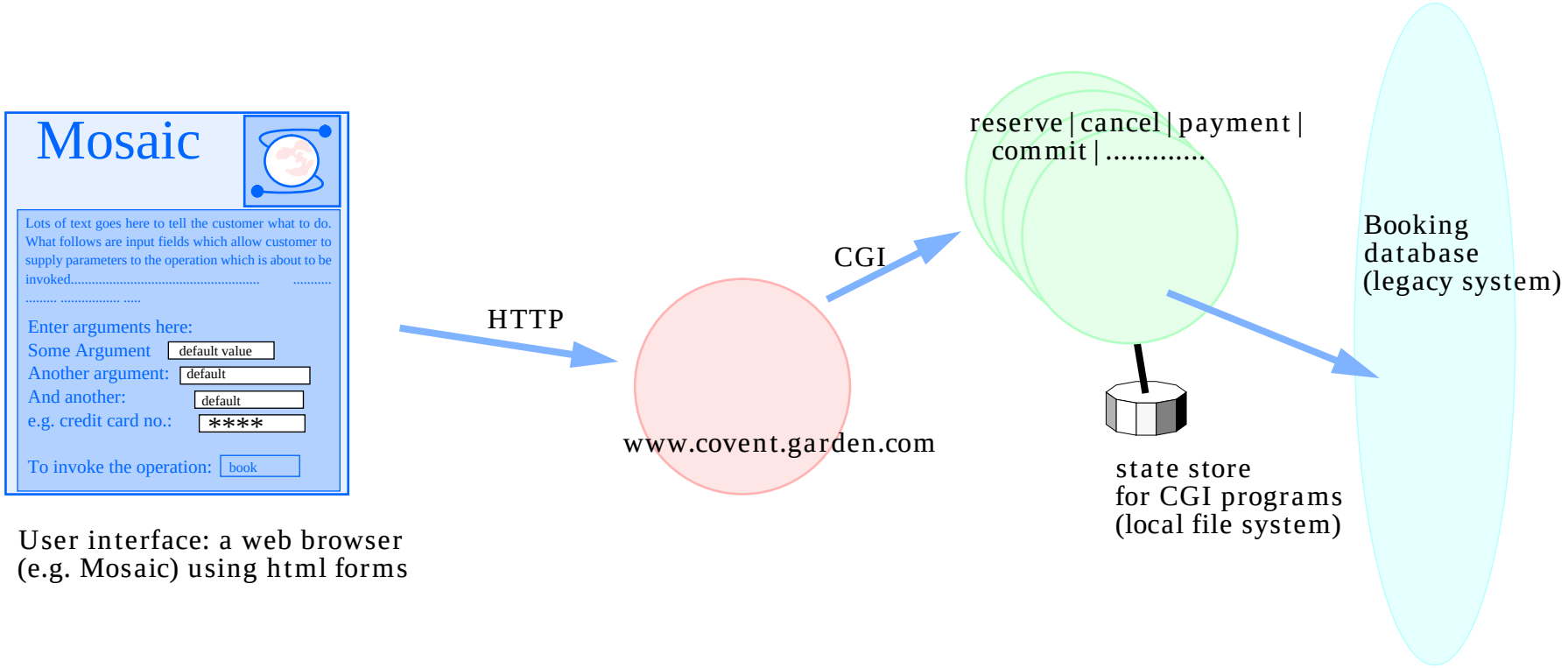

User interface: a web browser (e.g. Mosaic)

HTTP




Document store containing program information

# Extending the functionality of WWW using CGI Programs

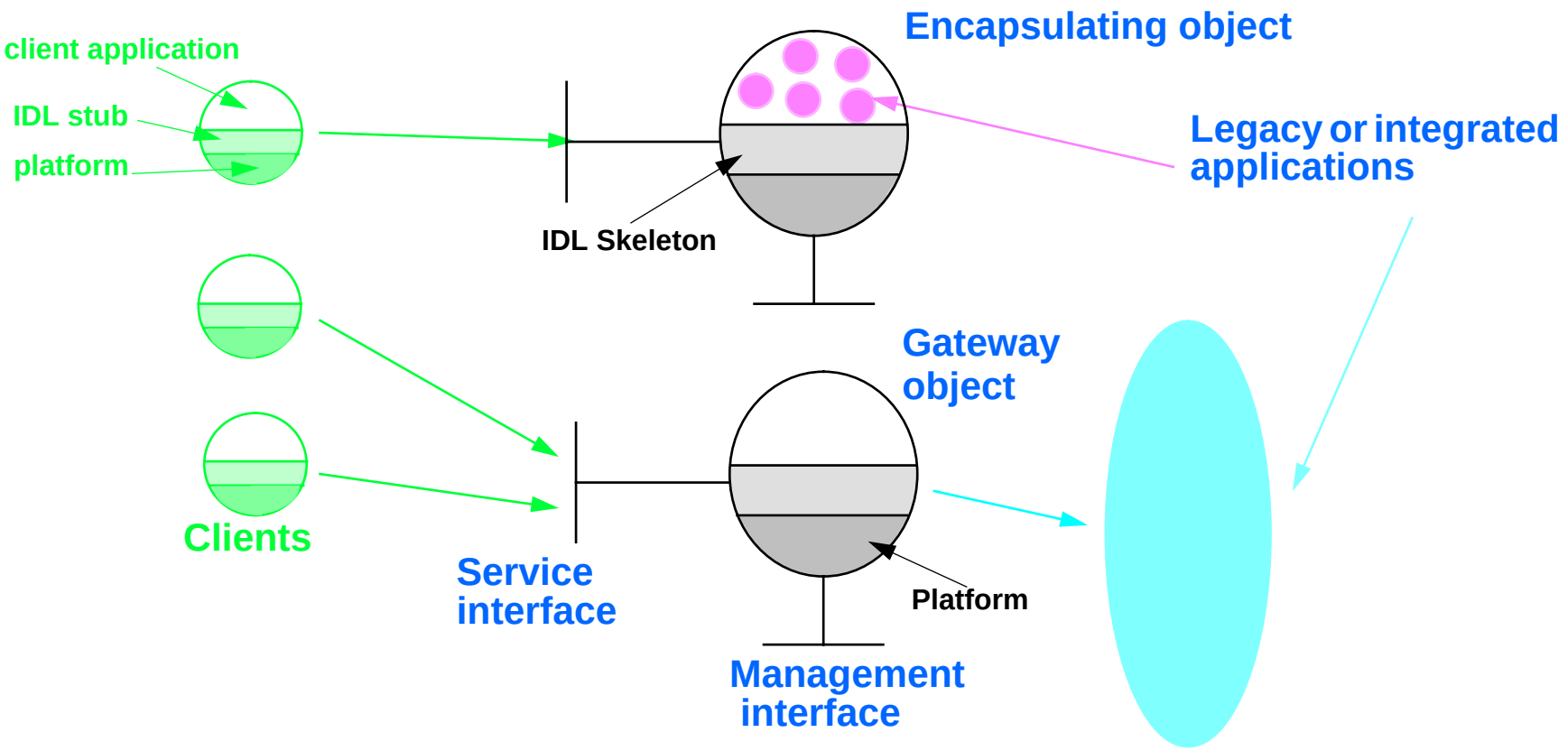



User interface: a web browser (e.g. Mosaic) using html forms


# Notes on the previous slide — not for presentation

- Currently server/browser interaction uses HTTP (HyperText Transfer Protocol) — an RPC protocol.
- The CGI (Common Gateway Interface) protocol running over (unix) pipes allows HTTP servers to communicate with other programs.
- This allows service providers to extend the functionality of their web servers (in ANSA we used it to provide a web interface to the trader).
- It is hard (error prone) to write the code to unmarshal the parameters to the CGI programs (no tool support).
- A CGI Program gets “forked” each time it is invoked, so state has to be stored externally.
- CGI programs are driven by HTML (HyperText Mark-up Language) forms technology. HTML forms are active documents: browsers marshal the parameters provided by the user and (using HTTP) passes them to the HTTP server. In turn, the HTTP server passes them to the CGI program.
- 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).
- If you can’t do what you want with an existing browser, implementing a new client requires programming on libwww. This is very difficult to do: libwww presents a complex API and is subject to change; programmers require a deep understanding of protocol details (HTTP) as well as the internals of the library

# The Solution: Distributed Object Technology




# Notes on the previous slide — not for presentation

- CORBA's (or distributed object technology) key value proposition is that it makes writing management and integration applications easy.
- The key idea is to access services through defined (in IDL) interfaces
- From the descriptions of the services we want to access, stub compilers can generate IDL stubs and skeletons which abstracts the programmer from the underlying protocols and API.
- Invocation of a remote service becomes very like local procedure call; it is no longer necessary to have a deep understanding of the underlying protocols or the internals of the platform.

# ANSAweb Phase 1: A Stub Compiler for the web


Browser

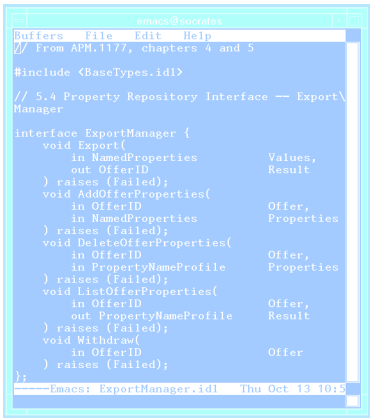
HTTP Server

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 1 — Benefits

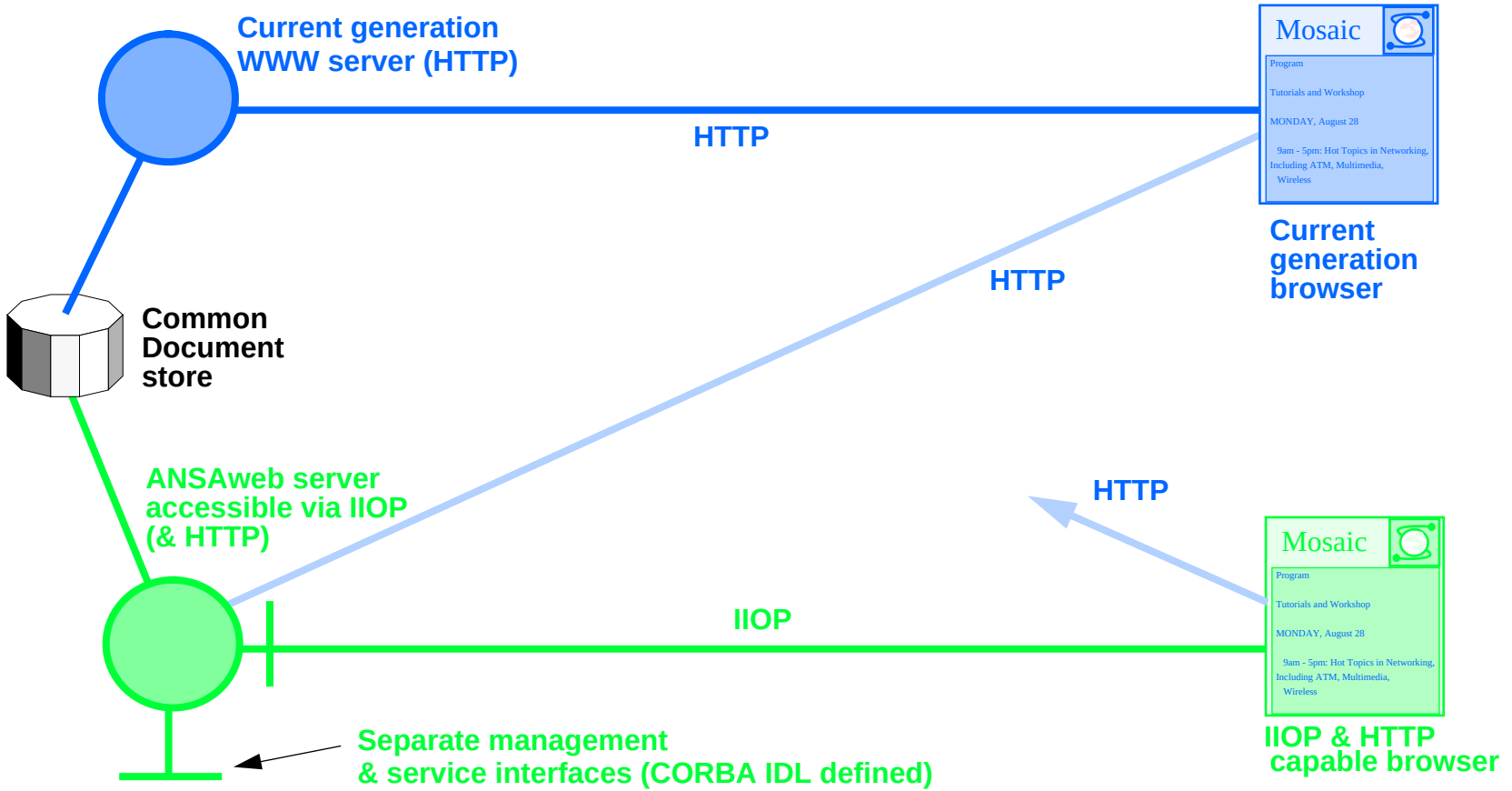
## The benefits arise from having to write less code

- **Productivity**
  - It is easier to interface existing systems into WWW
  - Do not require a deep understanding of platforms and protocols
  - Remote invocations look like local invocations
- **Less errors**
  - Template forms are correct HTML
  - Template forms, stubs and skeletons are consistent
- **Protection against changes**
  - Skeletons and stubs abstract the programmer from underlying platform — if the platform or protocol changes change the stub compiler and regenerate the stubs, do not have to rewrite the application.


# ANSAweb Phase 1 — Status

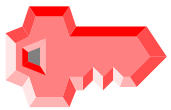
- **Public release on June 9 1995**
  - **Response to debate amongst WWW Consortium members**
  - **Establish “our turf”**
- **Release software runs on HPUX only**
  - **Ported to Solaris by France Telecom**
- **Early feedback — positive**

# ANSAweb Phase 2: Migrating the web to Distributed Objects



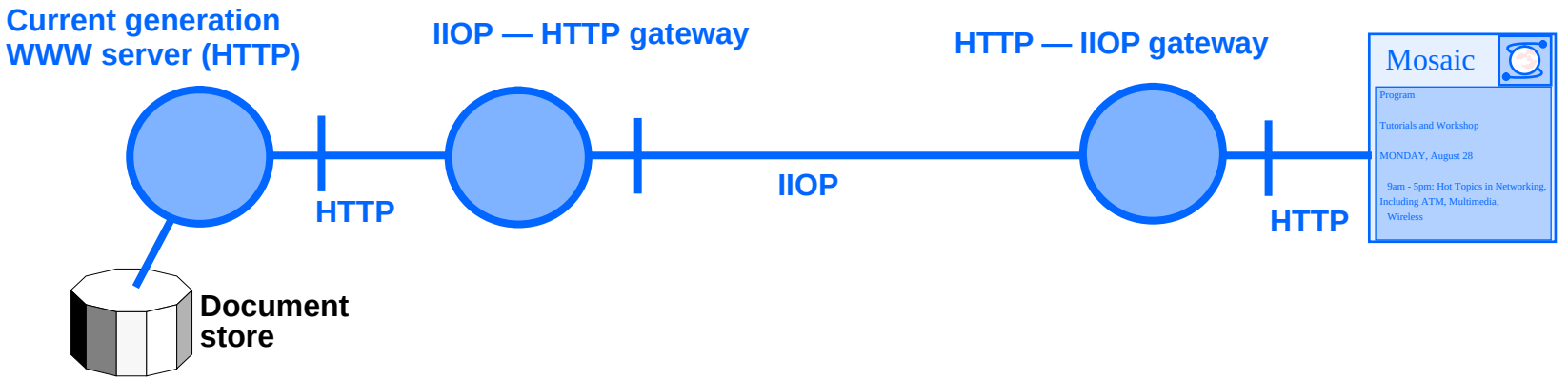

# The benefits

- **Performance of IIOP**
- **Extensibility**
  - **The ANSAweb server is essentially a CORBA object**
  - **The ANSAweb server gateway to the CORBA world**
  - **Interfaces defined and extensible in IDL**
  - **Integrating third party systems is a strength of CORBA**
- **Clients (browsers) migrate to object technology**
  - **Move away from the concept of the monolithic client which is difficult to upgrade**
- **Manageability**



**Backwards compatibility — no changes to HTML, preserve the http:// scheme, no new schemes (e.g. iiop://)**

# Getting to the promised land — the intermediate step



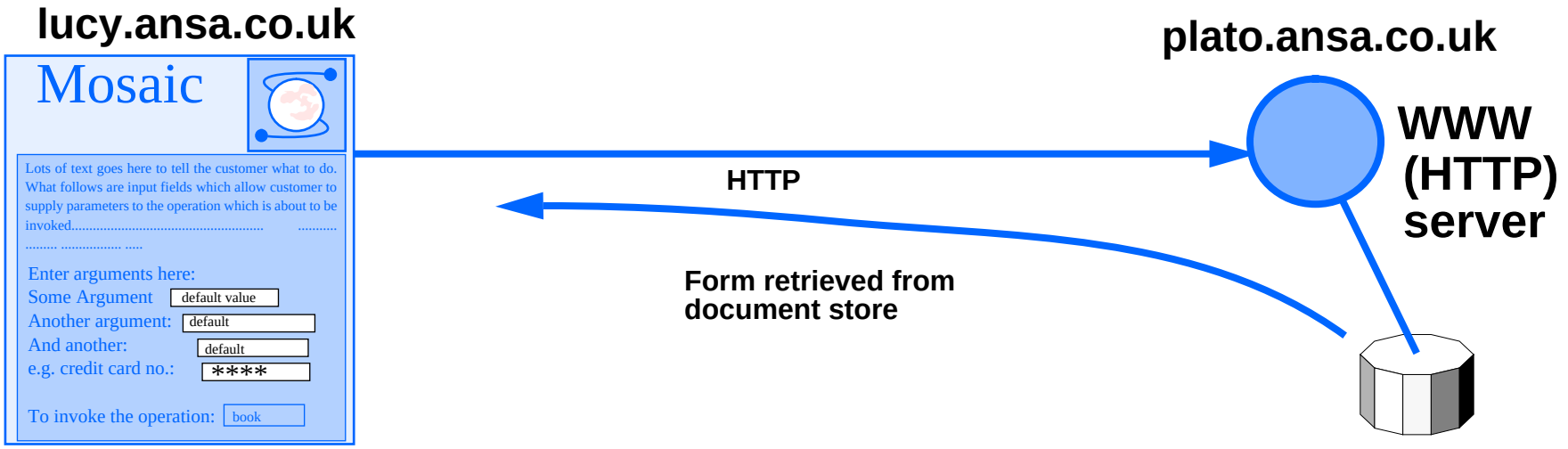

## Status

- Initial versions of the intermediate components working
- IIOP implementation is the Sun public domain code
- An IIOP stub compiler has been built using the Sun public domain front end


# Conclusions

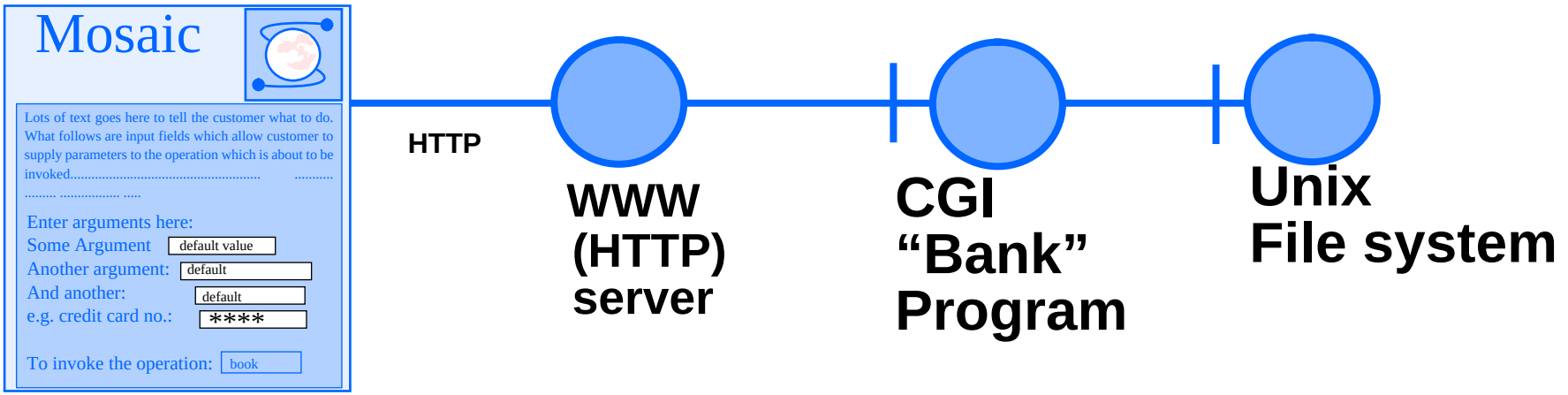
- The web is creating a uniform information space
- Some technology still required: efficient protocols, extensible front and backends, navigation tools, administration tools
- Distributed objects offers a solution
- The benefits include: programmer productivity; fewer programmer errors; hiding of changes in the underlying platforms and protocols; extensibility (ease of integrating new and existing systems); performance of the underlying protocols
- ANSAweb Phase 1 released June 9 1995 to establish a presence
- ANSAweb Phase 2 to be announced in December 1995 at the WWW conference

# The simple bank demonstration



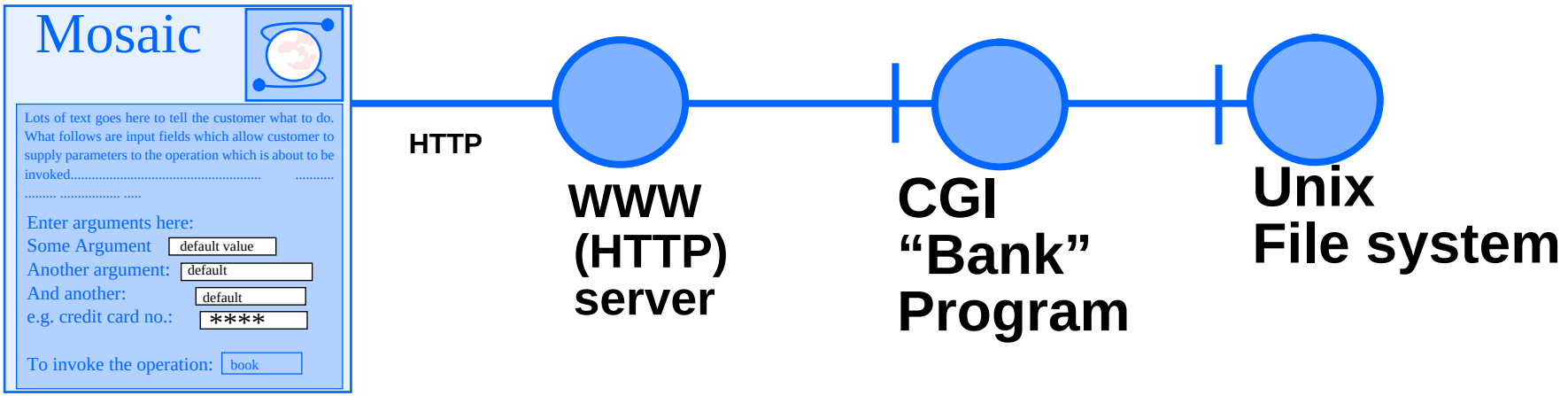

- The template form was generated by the stub compiler
- This was edited and loaded into the document store
- Comparison of template and edited form...

# The Simple Bank Manager




- The manager fills out the form interface to manage the bank
- HTTP server launches the bank program
- State is stored on the file system
- Result returned as HTML
- CORBA Exception Handling increases robustness

# The customer




- The customer fills out a form supplying PIN and Account number
- HTTP server launches the bank program
- Bank program checks the PIN and Account number are valid and generates a UI for the customer to access their account ONLY
- UI is a modified template form generated by stub compiler
- It is generated by the bank program on-the-fly, rather than being stored in the file system


# Lessons

- The application was developed in about 10 working hours
- 500 lines of C (includes an edited form generated by the stub compiler)
- Most of the complexity was in manipulating the unix file system to create a persistent state store and also setting file system locks to deal with concurrent access (most of the code was needed for this)
- Editing of forms takes only a few minutes
  - It is worth while running them through an HTML checker to make sure they are still correct after editing
- Experience suggests this is a useful tool for interfacing legacy systems into the web.