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**

Training

ANSAwise - Templates for Distributed Applications

Chris Mayers

Abstract

Distributed systems offer much to applications designers, but may be unclear how to partition the components of a conventional application. Apart from the business logic, two universal components are the user interface and the back-end database.

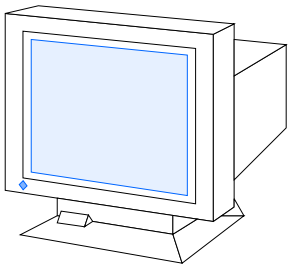
The module of the ANSAwise training programme introduces the most basic idea of client-server, as a stepping stone to the more general ODP approach covered in later modules.

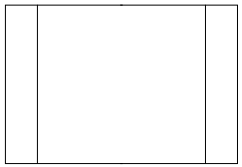
It also reinforces the importance of infrastructure, and explains how software integration platforms (SIPs) can help build systems for vertical markets.

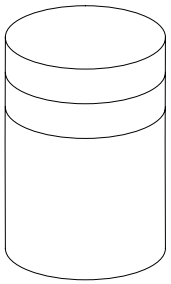
APM.1320.02 **Approved** 27th March 1996
Briefing Note

Distribution:
Supersedes:
Superseded by:

Templates for Distributed Applications







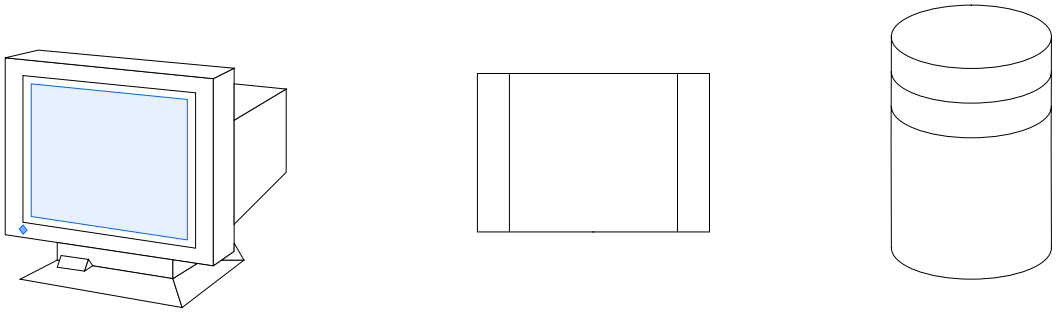
In this session

- Look at an approach to using LANs
  - sometimes called a ‘client-server’ application architecture
- Explain the limitations of some client-server technologies
- Show a technique for partitioning an application into components
  - and templates for deploying them
- Explain how a distributed systems approach embraces all of these

Two kinds of distributed systems templates

- client-server templates (horizontal)
- software integration platforms (vertical)
- ... generic “recipes” for system construction

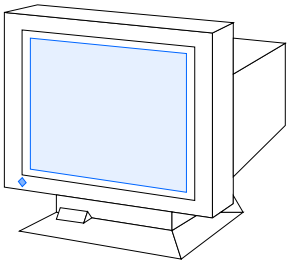
Basic systems components



Presentation Application Data

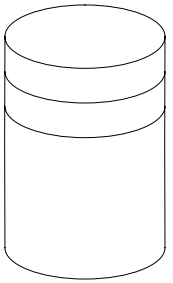
- **The desired system has the 3 traditional components:**
  - **Presentation: user interface**
  - **Application: business logic**
  - **Data: database or other long-term storage**

Functions of the presentation component



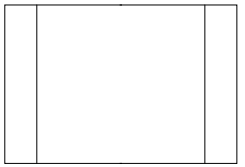
- **Provides the structure and representation of user interactions**
  - **dialogs and *user* process logic**
  - **device handling**
  - **device independence**
- **Supports usability and consistency across and between applications**

Functions of the data component



- **Storage and retrieval of information**

Functions of the application component



- **Executing the business logic of the application**

  - **the business rules**

Business rules

- **Business rules may be precise or heuristic (‘rules of thumb’)**
- **The rules automate decisions**
  - **formal decisions: “can this order ship by Thursday?”**
  - **policy decisions: <No single shipment can tie up more than 10% of the available inventory in stock for critical products>**
  - **resource management decisions: {Accepting orders only when inventory is in stock}**

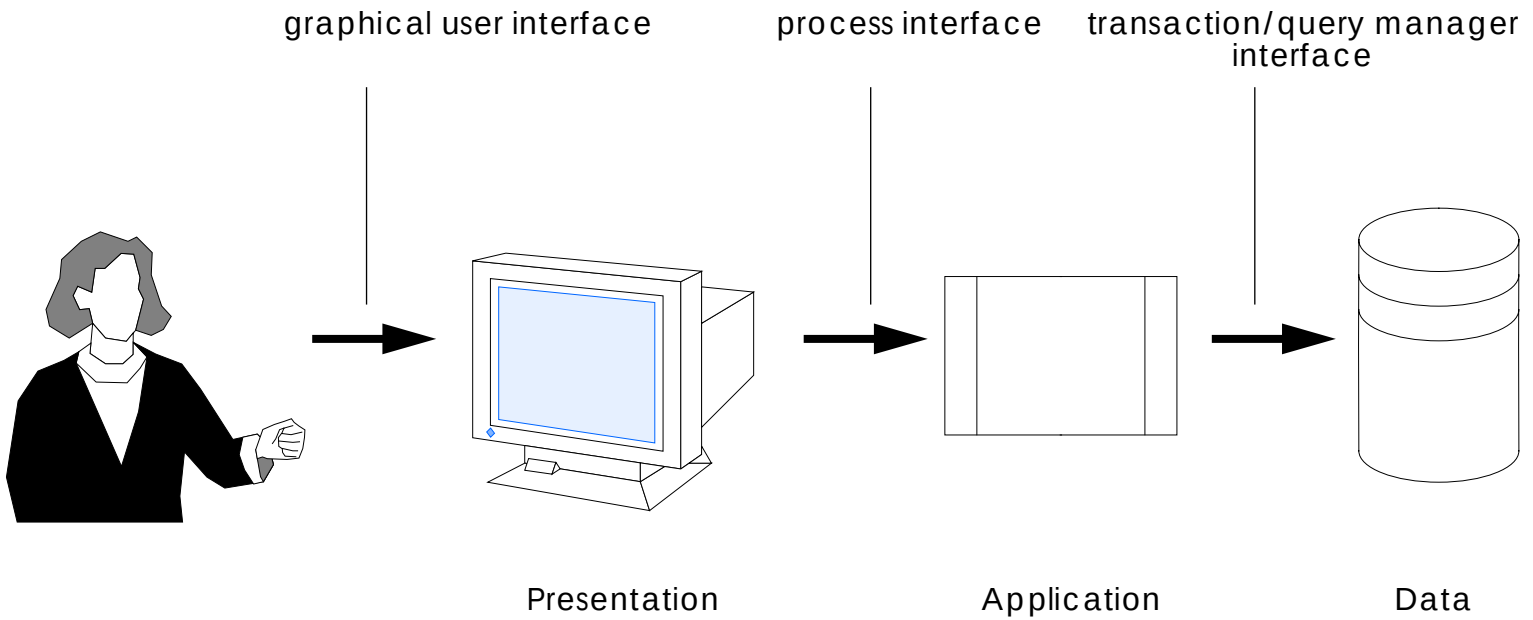
Separating the components

- **The presentation, application, and data components are different in nature**

- **Implementing each of them requires specialist skills**

  - **their design disciplines and implementation techniques are quite different**

Interfacing the components

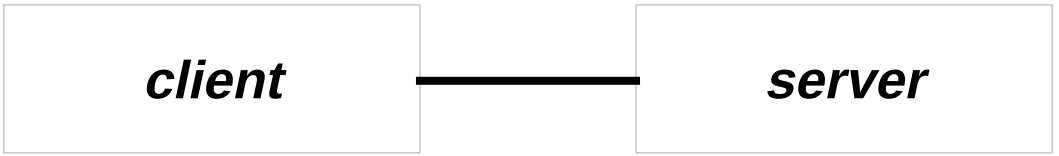


A question to ask your suppliers

- "Are these interfaces open interfaces"?
  - are the APIs portable?
  - are the protocols interoperable?
  - are specifications published and under vendor-neutral control?
  - are implementations available from more than one supplier?
- Different components may come from more than one source
  - some you buy, some you build yourself
- Also consider the tools (4GLs, GUI builders,...)
  - do they generate applications with these interfaces?

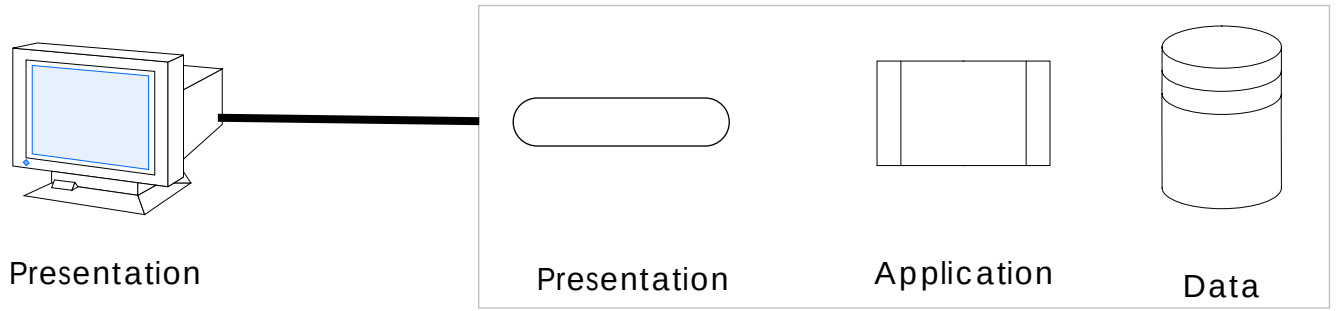
Deploying the components

- The components can be configured in several ways
  - Distributed Presentation
  - Remote Presentation
  - Distributed Logic
  - Remote Data Management
  - Distributed Database
- These partition the application between ‘client workstation’ and ‘server’ machine



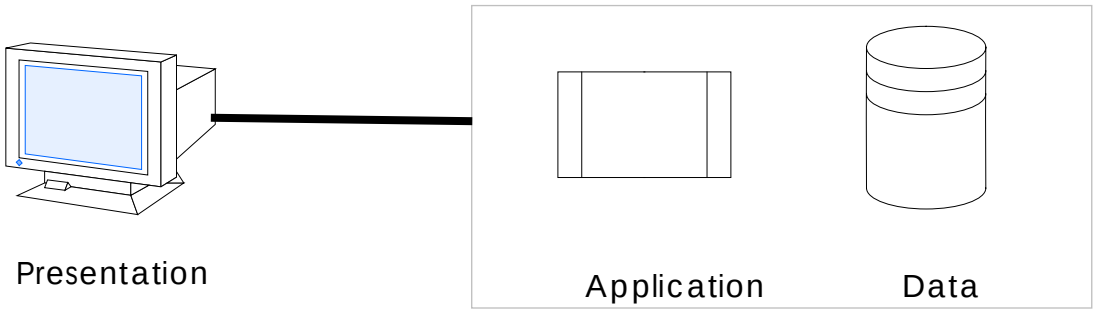
- Note: this is a restrictive idea of ‘client/server’

Distributed Presentation



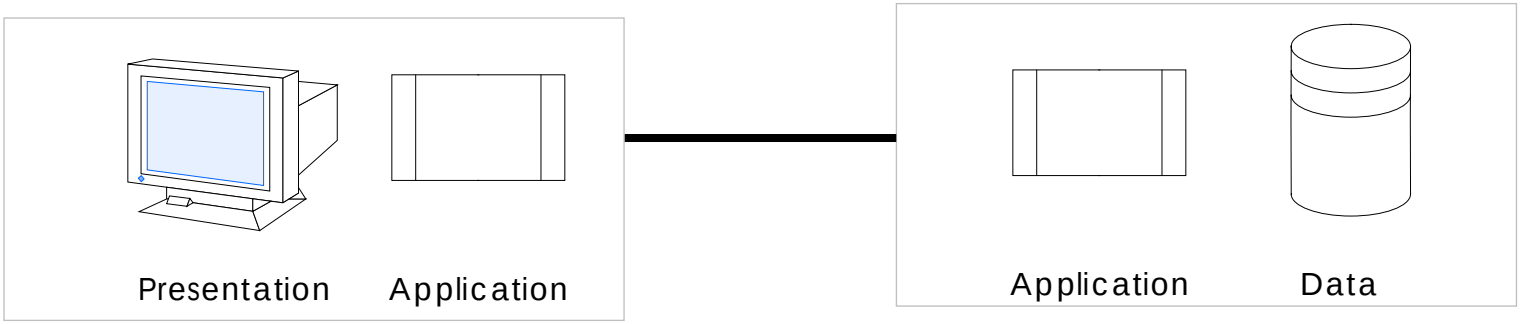
- Some of the presentation logic is in the server
  - For example, X Windows
- Little intelligence needed in the workstation display...
- ... but network traffic can affect user response times

Remote Presentation



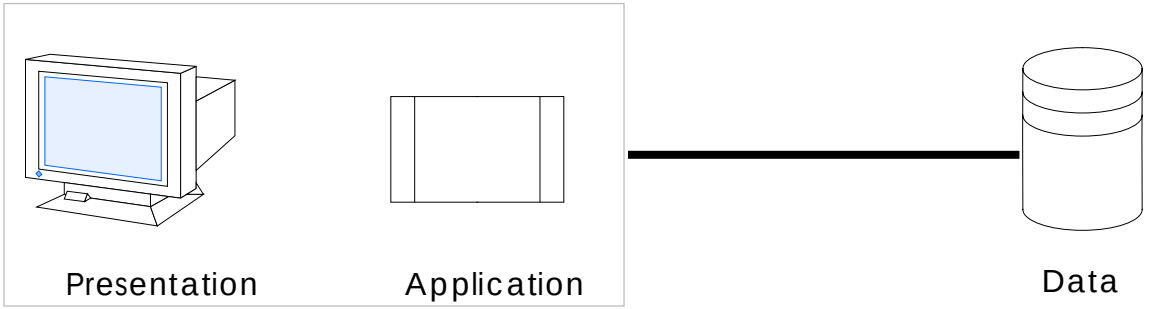
- **Exploits power of server**
  - **a database server with applications as stored procedures**
- **Reduces network traffic**
- **... but suffers from inflexible application development environment**

Distributed Logic

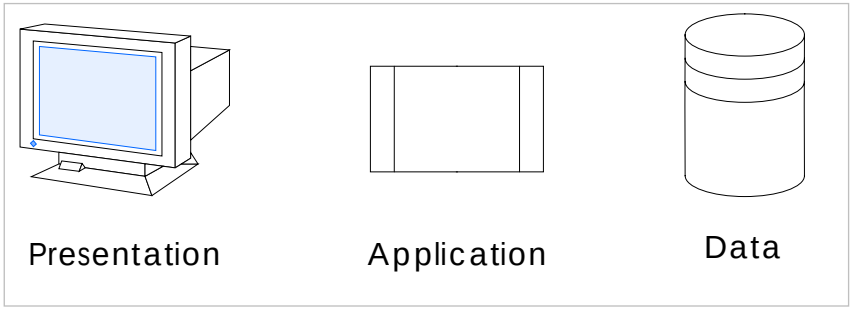


- **Some of the application logic is on the workstation**
- **Can be the most efficient in network traffic**
  - **but how is the application logic partitioned?**

Remote Data Management



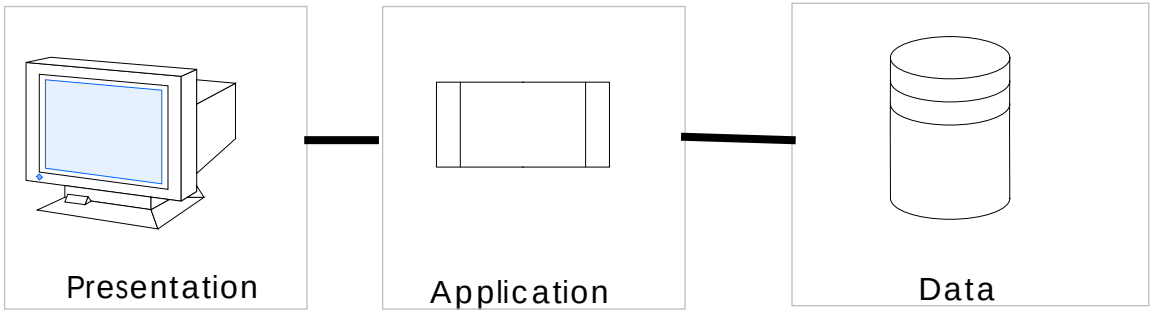
- **Exploits power of workstation**
- **... but it is difficult to manage the applications**
  - **for upgrades (many workstations having the same application)**
  - **to maintain data integrity**



Distributed Database

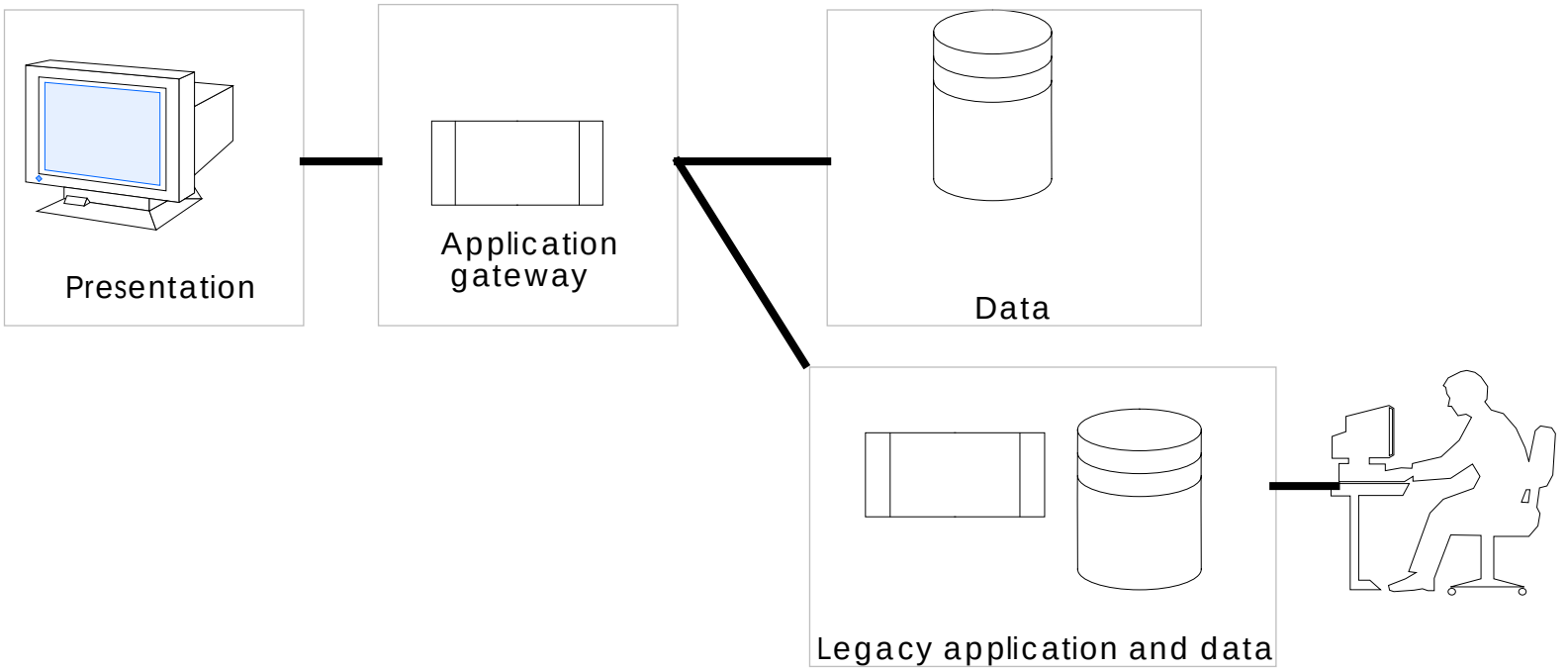
- Some of the database logic is on the workstation
  - multiple distributed databases can be supported
- ... but this has the disadvantages of remote data management
  - and distributed databases do not scale

Application server approach



- **Exploits power of workstation and server**
- **Application can be managed from a single point**
- **... the approach is the way to exploit Distributed Processing**

Application gateways for legacy systems



Small first-generation client-server systems

- **Typically...**
  - decision support
  - departmental or workgroup
  - 10-20 concurrent users
  - relational data only
  - read-only data
  - loosely integrated with existing systems
  - planned for a short life
  - using proprietary solutions
- **Often using the remote data management approach**
  - this approach does not scale to large systems

Large second-generation client-server systems

- **Typically...**
  - **operational support**
  - **enterprise-wide**
  - **100-1000 or more concurrent users**
  - **many data sources, including legacy systems**
  - **transactional, high integrity**
  - **tightly integrated with existing systems**
  - **planned for a long life**
  - **using open systems solutions**
- **The applications server approach is the most flexible**

What else is necessary?

- **Software infrastructure**
  - **operating systems**
  - **communications**
  - **standard building blocks**
- **Tools**

Application integration in specialist markets

- High-technology industries have a coordination problem
  - wide range of computer technology (including scientific instruments, field equipment)
  - ad-hoc use of data
  - world-wide teams needing the information this data could provide
- For example
  - Oil industry
  - Healthcare

Software integration platforms (SIPs) in vertical markets

- Major companies in these vertical markets
  - ... do not develop their own industry-specific software
  - ... but wish to integrate the systems they buy
- How can they ensure their suppliers' systems will interoperate?
  - by selecting systems built on a standardized software integration platform
  - standardized by industry-specific consortia

SIPs for rapid application creation

- SIPs aim at rapid application creation
  - rapid integration with customer's existing systems
  - rapid customization for customer-specific features
  - rapid configuration for delivery and deployment
- SIPs therefore emphasise the user interface, as well the distribution aspects
  - graphical tools for ease of customization and configuration
- SIPs allow niche players in vertical markets

Summary

- **Client-server templates and SIPs are each steps towards building more flexible applications**
- **Separating presentation, application, and data is important**
- **For more on these templates**
  - **on client/server strategies generally, see *Client/Server Strategies,* by David Vaskevitch (IDG Books)**