Chapter 11
APPLICATION ARCHITECTURE AND MODELING

Chapter Map

Irwin/McGraw-Hill Copyright © 2000 The McGraw-Hill Companies. All Rights reserved
Application Architecture

An application architecture specifies the technologies to be used to implement one or more (and possibly all) information systems in terms of DATA, PROCESS, and INTERFACE, and how these components interact across a network.

Realize that ALL of data, process, and interface is considered part of the application architecture – not merely the physical distribution of computers…

Think of application architecture as a set of construction blueprints that a building contractor might require for different building aspects such as framing, electrical, plumbing, ventilation, etc…

In this chapter we will concentrate on:

Objectives

• We will look at the degree to which information systems will be centralized or distributed. Consistent with modern trends, most new systems are distributed across a networks – including the intranet and Internet.

• We will look at the distribution of DATA across the network. (Most modern databases are either distributed or duplicated across networks, either in a client/server or network computing pattern)

• We will look at implementing technologies – which language and tools will be used?
• Will look at the technologies to be used to implement the user interface (inputs and outputs)
• Will look at the technologies to be used to interface with other systems.

• Differentiate between logical and physical data flow diagrams, and explain how physical data flow diagrams are used to model an information system's architecture.
• We will look at the tools first in this chapter.
Physical Data Flow Diagrams (DFDs)

By changing the Logical DFDs just a bit, LDFDs can communicate technical choices and other design decisions to those who will actually construct and implement the system. These then become Physical Data Flow Diagrams (PFDs).

Physical data flow diagrams (DFDs) model the technical and human decisions to be implemented as part of an information system.

Note that the process of creating a logical DFD and then a physical DFD still remain with us today.

Structure Charts may also be used.

Sample Physical Data Flow Diagram

Walk through this…
Physical Processes

• A physical process is either
  a processor, such as a computer or person, or
  a technical implementation of specific work to be
  performed, such as a computer program or manual
  process.

• Logical processes may be assigned to physical processors such as
  PCs, servers, mainframes, people, or devices in a network.

• A physical DFD would model that network structure.

• Each logical process requires an implementation as one or more
  physical processes. Note that a logical process may be split into
  multiple physical processes:
  • To define those aspects that are performed by people or computers.
  • To define those aspects to be implemented by different technologies.
  • To show multiple implementations of the same process.
  • To add processes for exceptions and internal control (e.g., security).

Comment:

• Some approaches to teaching these materials no longer teach a
  separate modeling technique for logical and physical networks
  because the industry and CASE tools never embraced a single
  physical modeling notation for networks.

• The authors suggest using physical DFDs to model networks.

• This approach seems to be quite effective.
Physical Process Notation

ID is often a number indicating a hierarchy

This is a typical use of a Process Box.

Also, a specific office is sometimes used to indicate ‘responsibility.’ (OPR)

These are normally found at ‘higher levels’ in DFDs.

This is the Gane and Sarson notation. The DeMarco and Yourdon notation does not provide for a separate symbolic differentiation between logical and physical processes. Depending on your choice of CASE or automated modeling tool, may required slightly different notations.}
Samples of Physical Processes

Showing physical implementations from the logical process.

<table>
<thead>
<tr>
<th>Logical Process</th>
<th>Sample Physical Process Implementations</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.3 Check Customer Credit</td>
<td>4.3 Check Customer Credit COBOL/CICS Visual Basic Quickbooks</td>
</tr>
</tbody>
</table>

Samples – but mutually exclusive

A Process may be implemented as:

- A purchased application software package
  - Also called commercial off-the-shelf (COTS) software

- A system or utility program

- An existing application program
  - May require modification

- A program to be written
Physical Processes – last overhead

- Note that the number of processes on a PDFD will almost always be greater than those in a LDFD because of implementation decisions.
- Maturing from ‘what’ to ‘how…’
- The final PDFD must reflect both manual and computer processes required as part of the implementation strategy.
Physical Data Flows

A physical data flow represents any of the following:

- The planned implementation of an input to, or output from a physical process.
- A database command or action such as create, read, update, or delete.
- The import of data from, or the export of data to another information system across a network.
- The flow of data between two modules or subroutines (represented as physical processes) in a program.
- Used to factor a program into modules – perhaps to pass parameters, variables, and arguments between modules of a program.
- This is a reminder that DFDs can be used as programming tools.

Sample Physical Data Flows

There are always alternative physical implementations for any logical data flow.
Sample Physical Data Flows (continued)

The diagram illustrates various logical data flows, their implementations, and the resulting physical data flows. The image highlights the importance of showing manual processes or forms within the data flow diagram to ensure a comprehensive understanding of the system's functionality.

External Agents and Data Stores

This section likely discusses the role of external agents and data stores in the context of systems analysis and design. It may cover how these elements interact with the system, providing a framework for data management and external interface considerations.
Physical External Agents and Data Stores

Physical external agents are carried over from the logical DFD models.

A physical data store represents the planned implementation of one of:
- A database
- A table in a database
- A computer file
- A tape or media backup of anything important
- A temporary file or batch
- Any type of noncomputerized file

Physical Data Store Notation

<table>
<thead>
<tr>
<th>ID (opt)</th>
<th>Implementation Method: Data Store Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID (opt)</td>
<td>Data Store Name (Implementation Method)</td>
</tr>
</tbody>
</table>
## Physical Data Store Implementations

<table>
<thead>
<tr>
<th>Logical Data Store</th>
<th>Implementation</th>
<th>Physical Data Store</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human Resources</td>
<td>A database (multiple tables)</td>
<td>Oracle: Human Resources DB</td>
</tr>
<tr>
<td>Marketing</td>
<td>A database view (subset of a database)</td>
<td>SQL Server: Northeast Marketing DB</td>
</tr>
<tr>
<td>Purchase Orders</td>
<td>A table in a database</td>
<td>MS Access: Purchase Orders</td>
</tr>
<tr>
<td>Accounts Receivable</td>
<td>A legacy file</td>
<td>VSAM File: Accounts Receivable</td>
</tr>
<tr>
<td>Tax Rates</td>
<td>Static data</td>
<td>ARRAY: Tax Table</td>
</tr>
<tr>
<td>Orders</td>
<td>An off-line archive</td>
<td>TAPB Backup: Closed Orders</td>
</tr>
<tr>
<td>Employees</td>
<td>A file of paper records</td>
<td>File Cabinet: Personnel Records</td>
</tr>
<tr>
<td>Faculty/Staff Contact Data</td>
<td>A directory</td>
<td>Handbook: Faculty/Staff Directory</td>
</tr>
<tr>
<td>Course Enrollments By Date</td>
<td>Archived reports</td>
<td>REPORT MGE: Course Enrollment Reports</td>
</tr>
</tbody>
</table>

Where can we use this information later?

- So, Physical DFDs use the same symbols as Logical DFDs

- Can use this physical model to design the internal and external details for each data store and data flow.
**INFORMATION TECHNOLOGY ARCHITECTURE**

- Technology is changing all the time.
- Systems Analysts must always learn new technologies:
  - Uses, application domains; restrictions.
  - Read trade journals

- Data, process and interface: key building blocks of all information systems.
- Any software design must address the application’s architecture, that is, the **distribution** of process, data, and interface.

- Nowadays, previous stand-alone mainframe and personal computers are being linked together in various ways using servers and forming very complex networks.
  - Processes are distributed; data is distributed; users are world-wide…
  - Consequently, processes, data, and interfaces are similarly distributed or at least duplicated over these complex networks.

---

**Distributed versus Centralized Systems**

Today’s systems are not monolithic, mainframe-based systems. They are built on some combination of networks to form distributed systems.

A **distributed system** is one in which the DATA, PROCESS, and INTERFACE components of an information system are distributed to multiple locations in a computer network.

-> Processing workload is **distributed** across the network.

In **centralized systems**, a central, multi-user computer hosts all the DATA, PROCESS, and INTERFACE components of an information system.

-> Users interact with the system via terminals (or terminal emulators).
Why Distributed Systems?

- Modern business systems: already decentralized and distributed.
- Distributed computing moves information and services closer to the customers and users who need them.
- Distributed computing consolidates the power of personal computers across the enterprise.
- Distributed computing solutions are more user-friendly because they utilize the PC as the end user interface.
- Personal computers and network servers are cheaper than centralized mainframe computers.
  - This may be controversial. Some experts sincerely believe that distributed computing is more costly to maintain. But this belief has not deterred the proliferation of distributed solutions.

Distributed Computing Systems Architectural Layers

- **Presentation layer** — the user interface
  - the presentation of inputs and outputs to the user
  - Presentation layer logic—such as input editing
    - the processing necessary to generate the presentations.

- **Application logic layer** — the business rules, policies, and procedures
  - includes the logic and processing procedures; the computations.

- **Data manipulation layer** —
  - includes the commands and logic to store and retrieve data to and from the database

- **Data layer** — the actual business data itself.

In some books, the presentation and presentation logic layers are combined.
Flavors of Distributed Computing

Discuss:

Various combinations

Let's look closely at the three primary 'flavors' of distributed systems architecture.

File Server Architecture

A local area network (LAN) is a set of client computers (PCs) connected to one or more server computers either through cable or wireless connections over relatively short distances.

A file server system is a LAN-based solution in which a server hosts only the data layers of an information system. All other layers are implemented on the client computers. Disadvantages include:

- Frequently excessive network traffic to transport data between servers and clients.
- Client must be fairly robust ("fat") because it does most of the work
  - all applications software loaded on client
- Database integrity can be easily compromised.
  - In use, entire file / table of records must be downloaded to client for data manipulation logic to do its thing. (lockout!) Server only contains data. No data manipulation. Done on client.
  - performance degrades with number of simultaneous users.
- Very popular for small groups; File server databases like Access can be used to construct prototypes for more robust client/server architectures.
Many students think Access applications that store their data on a file server are truly client/server applications (and many books proliferate the misconception).

Previous slide, when contrasted with the client/server equivalent slides, is intended to dispel that myth.

Because the client executes any CREATE, READ, UPDATE, and DELETE commands for a file-server database (such as Access), the entire table must be locked out from other users and transported to the client to execute the command.

Result is increased data traffic on the network, and slow performance since the table must be returned and unlocked before the next user can do anything with it.
Client/Server Architecture—The Clients

A **client/server system** is a solution in which the presentation, presentation logic, application logic, data manipulation, and data layers are **distributed** between client PCs and one or more servers.

A **thin client** is a personal computer that **does not** have to be very powerful (or expensive) in terms of processor speed and memory because it only presents the user interface to the user.

A **fat client** is a personal computer or workstation that is typically more **powerful** (and expensive) in terms of processor speed, memory, and storage capacity.

Most PCs are considered to be fat clients.

– We are not using “fat client” in a negative sense.

– For the time being, most workers require fat clients for their personal productivity applications (e.g., word processing and spreadsheets).

– But they **may use thin client features** on those fat clients (e.g., a Web browser or terminal emulator) to run or access remote applications (such as those provided by the information systems unit of the company).
Client/Server Architecture—The Servers – various…

- A database server hosts one or more shared databases but also executes all data manipulation commands.
  - examples: Oracle, Microsoft SQL Server, IBM Universal Database

- A transaction server hosts services that ultimately ensure that all database updates for a single transaction succeed or fail as a whole.
  - e.g. CICS, Microsoft Transaction Server

- An application server hosts the application or business logic and services for an information system.
  - must communicate with front end with clients for presentation and back end with database server. typical technologies: CORBA, or COM+

- A messaging or groupware server hosts services for e-mail, calendaring, and other work group functionality.
  - typical technologies: Lotus Notes and Microsoft Exchange Server

- A web server hosts Internet or intranet web sites and services, communicating through thin-client interfaces such as web browsers.
  - typically use html for documents, xml for data formats.
These server categories are **not** mutually exclusive.

For example, many database server and transaction server functions may be integrated on a single physical server.

Also, web and messaging server functions are being integrated with each new release of some products.

---

**Client/Server—Distributed Presentation**

A **distributed presentation client/server system** is a solution in which the presentation and presentation logic layers only are **shifted** from the server to reside on the client.

Application logic, data manipulation, and data layers remain on the server (frequently a mainframe).
**Client/Server—Distributed Presentation**

Typically the old CUI (character user interface) are taken from legacy apps and regenerated as GUIs that run on a PC. (only UI is distributed)

Adv1: can be implemented quickly; Most elements of legacy apps remain unchanged.

Adv2: Users get a welcome, familiar interface via a GUI.

Adv3: Extends useful lifetime expectancy of legacy apps.

Dis1: Applications functionality is not significantly improved.

CASE tools called screen scrapers read the CUI and generate a first-cut GUI that can be modified by a GUI editor.

This is sometimes called “the poor man’s client/server.”

In reality, the underlying technology allows organizations with constrained resources to give mainframe systems a client/server facelift that extends their useful lifetime until they can be redesigned to more contemporary architectures.
A **distributed data client/server** system is a solution in which the data and data manipulation layers are placed on the server(s), and the application logic, presentation logic, and presentation layers are placed on the clients.

This is sometimes called **two-tiered client/server** computing.

Again, lots of flavors

This architecture characterizes most Visual Basic and Powerbuilder applications in existence today.
Differences between file server system and distributed data client/server system

- Both use a database server for data storage.
- But: client-server’s server also **executes** all data manipulation logic.
  - all application logic, presentation logic and presentation services are handled by the client machine.
- In file server, **everything other than data storage** is done on the client machine.
Advantages / Disadvantages of two-tier client-server architectures

- **Client**: Much less network traffic, since only database requests and records are transmitted over lines.
- Only records that are being used by the client need to be locked on the server. (Simultaneous usage ok – except for record update).
- **BUT**, clients need to be reasonably ‘fat’ because the clients must perform all the application logic.
- Clients normally perform their functions with client/server programming language (e.g., VB or Visual C++, whose object code must run on the client. (Sometimes common, stored business procedures may be stored on the server, but this is controversial.)
- The database server stores the database and executes the data manipulation commands. Clients merely send their instructions to the server. Servers may return command processing results. (Oracle and Microsoft SQL Server do this).
- **Key disadvantage**: application logic must be duplicated / maintained on all clients. Then version control / upgrades may be an issue.

Client/Server—Distributed Data and Application (n-tiered)

A distributed data and application client/server system is a solution in which: (1) the data and data manipulation layers are placed on their own server(s), (2) the application logic is placed on its own server, and (3) the presentation logic and presentation layers are placed on the clients.

This is sometimes called **three- or n-tiered client/server computing**. It requires **design partitioning**.

**Partitioning** is the art of determining how to best distribute or duplicate application components (DATA, PROCESS, and INTERFACE) across the network.

This is the architecture of your projects.
Client/Server—Distributed Data and Application

- Two-tiered systems cannot handle high number of clients. (All application logic executes on each client!)
- Three-tiered systems have the application logic / transaction server on its own box.
- Three-tiered client/server logic can be written and partitioned across multiple servers using languages such as VB and C++
- In an n-tiered architecture, clients execute very little – usually the user interface and some real stable personal applications logic. Simplifies client configuration.

- **Biggest problem: complexity in design and development. Biggest problem in here is ‘partitioning.’**
Internet- and Intranet-based Architectures (Network Computing)

A network computing system is a multi-tiered solution in which the presentation and presentation logic layers are implemented in client-side Web browsers using content downloaded from a Web server. The presentation logic layer then connects to the application logic layer that runs on the application server, which subsequently connects to the database servers on the backside of the system.

The greatest potential of this approach is its applicability to redesign of traditional information systems to run on an intranet. An intranet is a secure network, usually corporate, that uses Internet technology to integrate desktop, work group, and enterprise computing into a cohesive framework.

The authors feel that ‘network computing’ is a fundamental shift away from what they just described as client/server. They call this architecture internet-based computing architectures – and that these are shaping the information systems of most businesses.

This is the architecture that is driving e-commerce and e-business.

Expect this architecture to eventually replace most two- and three-tier solutions.
Internet- and Intranet-based Architectures

![Diagram of Internet- and Intranet-based Architectures]

Internet- and Intranet Technologies

- **Java**
  - Mostly for programming server-side application logic called “servlets”
  - Occasionally for programming client-side application logic called “applets”
- **HTML (HyperText Markup Language)**
  - Mostly for programming the presentation layer
- **XML (Extensible Markup Language)**
  - Mostly for programming data content to be transported across the web
- **SQL (Structured Query Language)**
  - Universal standard language for database manipulation
- **Web Browsers**
Data Architectures – Should be quite familiar to you…

A **relational database** stores data in tabular form. Each file is implemented as a table. Each field is a column in the table. Related records between two tables are implemented by intentionally duplicated columns in the two tables.

A **distributed relational database** distributes or duplicates tables to multiple database servers located in geographically important locations.

A **distributed relational database management system** is a software program that controls access to and maintenance of stored data in the relational format.

---

Types of Data(base) Distribution

**Data partitioning** truly **distributes** rows and columns of tables to specific database servers with **little or no duplication** between servers.

- Vertical partitioning assigns different columns to different servers.
- Horizontal partitioning assigns different rows to different servers.

**Data replication** **duplicates** some or all tables (or parts of tables) on more than one database server. Database technology controls access to, and manages consistency of duplicated data across the servers.
### Data Partitioning versus Data Replication

<table>
<thead>
<tr>
<th>Logical Data Store</th>
<th>Physical Data Stores using Partitioning</th>
<th>Physical Data Stores using Replication</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUSTOMERS</td>
<td>Oracle 7: REGION 1 CUSTOMERS</td>
<td>Not applicable. Branch offices do not need access to data about customers outside of their own sales region.</td>
</tr>
<tr>
<td>PRODUCTS</td>
<td>Oracle 7: REGION 2 CUSTOMERS</td>
<td>Oracle 8i: PRODUCTS (Master)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Oracle 8i: PRODUCTS (Replicated Copy)</td>
</tr>
</tbody>
</table>

- Not applicable. Branch offices do not need access to data about customers outside of their own sales region.

### Interface Architectures: Inputs, Outputs, Middleware

-