Class Announcements

- **Midterm Tuesday 5/5**
- Study guide posted
Architecture Example
Time sharing

ASCII terminal
(no graphics)

Point-to-point wire
(no network)

Mainframe
(database and
application server)
Two-tier client/server

Local-area network

Server/Mainframe
Three-tier client/server

Client → Application server → Enterprise data server

Diagram showing a three-tier client/server architecture with a client computer, an application server, and an enterprise data server connected by arrows indicating data flow.
System integration

1. Architecture
2. subsystem implementation
3. system integration
   Bring together subsystems and make them achieve desired system functionality
   - Testing
   - Modifications often needed
Emergence

Subsystems are
- specialized
- have simple functionality

Higher-level system functionality arises from the interaction of subsystems
Called: Emergence

e.g. airplane flies, but subsystems can’t
Why system decomposition?

- Divide and conquer approach to containing complexity
- Reuse
- Consonant with industry structure (unless system is to be supplied by one company)
- Others?
Networked computing infrastructure

by
David G. Messerschmitt
Layering

Elaboration or specialization

Services

Existing layers
Example of Layering: networking

Application

Transport

Network

Link

Physical

Messages

Packets

Frames

Bits

Signals
Software Layering
Operating system functions

- Graphical user interface (client only)
- Hide details of equipment from the application
- Multitasking
- Resource management
  - Processing, memory, storage, etc
- etc
Middleware Functions

- Capabilities that can be shared by many applications, but that is not part of OS
  - Example: Database Management System (DBMS)
- Hide details of OS from application
  - Java Virtual Machine

- More purposes we’ll talk about later.
What’s a database?

Database

- File with specified structure
- Example: relational table
## A Database

<table>
<thead>
<tr>
<th>Year</th>
<th>City</th>
<th>Accommodation</th>
<th>Tourists</th>
</tr>
</thead>
<tbody>
<tr>
<td>2002</td>
<td>Oakley</td>
<td>Bed&amp;Breakfast</td>
<td>14</td>
</tr>
<tr>
<td>2002</td>
<td>Oakley</td>
<td>Resort</td>
<td>190</td>
</tr>
<tr>
<td>2002</td>
<td>Oakland</td>
<td>Bed&amp;Breakfast</td>
<td>340</td>
</tr>
<tr>
<td>2002</td>
<td>Oakland</td>
<td>Resort</td>
<td>230</td>
</tr>
<tr>
<td>2002</td>
<td>Berkeley</td>
<td>Camping</td>
<td>120000</td>
</tr>
<tr>
<td>2002</td>
<td>Berkeley</td>
<td>Bed&amp;Breakfast</td>
<td>3450</td>
</tr>
<tr>
<td>2002</td>
<td>Berkeley</td>
<td>Resort</td>
<td>390800</td>
</tr>
<tr>
<td>2002</td>
<td>Albany</td>
<td>Camping</td>
<td>8790</td>
</tr>
<tr>
<td>2002</td>
<td>Albany</td>
<td>Bed&amp;Breakfast</td>
<td>3240</td>
</tr>
<tr>
<td>2003</td>
<td>Oakley</td>
<td>Bed&amp;Breakfast</td>
<td>55</td>
</tr>
<tr>
<td>2003</td>
<td>Oakley</td>
<td>Resort</td>
<td>320</td>
</tr>
<tr>
<td>2003</td>
<td>Oakland</td>
<td>Bed&amp;Breakfast</td>
<td>280</td>
</tr>
<tr>
<td>2003</td>
<td>Oakland</td>
<td>Resort</td>
<td>210</td>
</tr>
<tr>
<td>2003</td>
<td>Berkeley</td>
<td>Camping</td>
<td>115800</td>
</tr>
<tr>
<td>2003</td>
<td>Berkeley</td>
<td>Bed&amp;Breakfast</td>
<td>4560</td>
</tr>
<tr>
<td>2003</td>
<td>Berkeley</td>
<td>Resort</td>
<td>419000</td>
</tr>
<tr>
<td>2003</td>
<td>Albany</td>
<td>Camping</td>
<td>7650</td>
</tr>
<tr>
<td>2003</td>
<td>Albany</td>
<td>Bed&amp;Breakfast</td>
<td>6750</td>
</tr>
</tbody>
</table>
Storage Middleware example: DBMS

- Database Management System (DBMS)
  - Manage Multiple databases
  - Allow multiple applications to access common databases
  - Implement standard data “lookup” (query) functions.
Client - Server Computing
Client Server Example

Client

“I want to see www.google.com”

Server

<html><head><meta http-equiv="content-type" content="text/html; charset=UTF-8"><title>Google</title><style><!--body,td,a,p,.h{font-family:arial,sans-serif;}
.h{font-size: 20px;}
.q{color:#0000cc;}
//--> ...
Client Server Example - Layers Revealed

Client Application:

Server Application:

Internet

Packet

Packet

Packet

Packet
3-Tier Client Server Architecture example

Client

Application Server

What is Bob’s balance?

$0.50

Shared data

Clicks, keystrokes

Client

Balance $0.50

What is Bob’s balance?

$0.50

Shared data

Clicks, keystrokes
3-Tier Client Server Architecture example
3-Tier Client Server Architecture example

- Client
- Application Server
  - Web Server
  - Application Logic
  - Database Management System (DBMS)
  - Database

What is Bob’s Balance?
3-Tier Client Server Architecture example

- **Client**
- **Application Server**
  - **Web Server**
  - **Database Management System (DBMS)**
    - **Database**

**What is Bob’s Balance?**

In some implementations, Application Logic and Web Server can be put on Different machines.

**Shared data**
### Relational Database

<table>
<thead>
<tr>
<th>Customer</th>
<th>Balance</th>
<th>Customer Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>$527</td>
<td>Silver</td>
</tr>
<tr>
<td>Bob</td>
<td>$0.50</td>
<td>Bronze</td>
</tr>
<tr>
<td>Charles</td>
<td>$1000000</td>
<td>Gold</td>
</tr>
</tbody>
</table>
DBMS Responsibilities

- Hide Changes in the Database hardware from the Application

- Standard operations on the data, including searches, such a search is called a **query**.

- Separate Database Management from Applications, so that many applications can access the same data.

- Security, Integrity, Backup, fault tolerance, etc.
3-Tier Client Server Architecture in General

Client:
- Accept instructions from user
- Make requests of server
- Display responses of server

Application Server:
- Takes inputs from client
- Decides what to be done next
- Decides what shared data to access and manipulates it
- Processes shared data

Shared data:
- Support multiple applications with common data
- Protect critical data
- Decouple data administration and application administration
How do you begin to architect a solution for a problem like this?

Break it into modules!
- New in-flight seatback system
  - Sell upgrades and seat swaps
    - (People who want to get away from sick people ...)
  - More legroom
  - Offer to exchange seats
Architecture

Wireless Link

Seat back devices

Wireless Link

servers

HEADQUARTERS

Airline Dataserver
When a module is composed of sub-modules, the architecture is **hierarchical**.
We also make use of layers
Granularity tradeoff.

- How big should we make the modules
  - Many simple small ones
  - Or a few complicated big ones...

- This aspect of modularity is called *granularity*.

- Which is better?
In-plane Server

- Again, we see layering and hierarchy.
- Between each module we specify an **interface**

**Server Application**

- **LinuxOS**

**Networking Infrastructure**

**Computation of key statistics**

**Communication With seat backs**

**Communication with airline database**

Standard Database “queries” (SQL) relayed to DBMS via OS and infrastructure
Our architecture makes use of the Existing interface of the airline database, so we don’t need to redesign it!
A simple interface: from within Architecture

- List of numbers
- Computation of key statistics
  - Mean, Variance

- HHC Application
  - Linux
  - Networking Infrastructure

- Communication with airline database
  - Communication with HHC

- Compute Mean and Variance
Interface specifications are often made precise by using **data types**.

- **Example type: float**
  - A number with a decimal place
  - Has a certain allowable range, and precision.
More on Data types

Data passing an interface is often specified in terms of a limited number of standard **data types**

Data type = range of values and allowable manipulation

Data type does not presume a specific representation, to allow heterogeneous platforms

- Representation must be known when data passes a specific module interface
Example data types

Integer
- "natural number between -32,767 and +32,768"
- Could be represented (in many ways) by 16 bits
  - since $2^n = 65,536$

Float
- "number of the form $m \times 10^n/32768$, where $m$ is in the range -32,767 to +32,768 and $n$ is in the range -255 to +256"
- Could be represented by 16+8 = 24 bits
More data types

Character

- “values assuming a-z and A-Z plus space and punctuation marks”
  - could be represented by 7 or 8 bits

Character string

- “collection of n characters, where n is customizable”
  - could be represented by 7*n bits
Compound data types

Programmer-defined composition of basic data types

Example:

```java
Employee {
    String name;
    String address;
    Integer year_of_birth;
    etc.
}
```
Interfaces

Computation of key statistics

PARAMETERS

N numbers of Float type

INTERFACE

Compute Mean and Variance

RETURNS

2 Numbers of float type that signify: Mean, Variance
**Implementation**

- One module should not be concerned with other module’s implementation
  - → “Separation of concerns.”
- One module should see the other only through its interface - implementation details hidden.
  - → Abstraction

**Computation of key statistics**

- **Module A**
  - Computation
- **Module B**
  - Compute Mean and Variance

**Formulae:**

\[
\text{MEAN} = \frac{1}{N} \sum_{i=1}^{N} x_i
\]

\[
\text{VARIANCE} = \sum_{i=1}^{N} (x_i - \text{MEAN})^2
\]
Implementation

Computation of key statistics

Module A

Module B

Compute Mean and Variance

$X_i, i=1..N$

Mean, Variance

Implementation 2:

$$\text{SUM} = \sum_{i=1}^{N} x_i$$

$$\text{MEAN} = \frac{\text{SUM}}{N}$$

$$\text{VARIANCE} = \sum_{i=1}^{N} (x_i - \text{MEAN})^2$$

- Though different, this implementation is ok too.

- We can choose the implementation details however we want, as long as we comply with the agreed interface.
Implementation

Computation of key statistics

Module A

Module B

Compute Mean and Variance

Implementation 1:

\[
\text{SUM} = \sum_{i=1}^{N} x_i
\]

\[
\text{MEAN} = \frac{\text{SUM}}{N}
\]

\[
\text{VARIANCE} = \sum_{i=1}^{N} (x_i - \text{MEAN})^2
\]

“Should he use it?”

- NO!!!! Why??

- Either A should compute “SUM” himself, or sit down with B and redesign the interface.
Encapsulation

The designer of B might take measures to hide “SUM” from A so that A is not able to violate the agreed interface.

Example: B does not declare “SUM” as a global variable.

Making a modules implementation details inaccessible to other modules is called **encapsulation**.
This simple interface example allows for only one action of module B.

- Action is “Compute mean and variance.”

Other examples are possible.
Possible software interface

Menu of actions

Example:

Action 1: Compute mean
Action 2: Compute variance
Action 3: Compute mode
Etc..
In addition to atomic actions, an interface may define protocols

- **Protocol** == finite sequence of actions required to achieve a higher level function

- One action can be shared by multiple protocols

- Multiple modules may participate in a protocol
Another Interface Example: Automatic teller machine (ATM)

What is the interface between this machine and the customer?
Steps

Define available actions

Define, for each higher level function, a protocol

- Single action or a finite sequence of actions
Interface building blocks

Message on screen or printed
- Menu of actions or returns from an action
- Touch selection of action

Keypad
- Input parameters to an action

Card reader
- Authentication, input parameters

Money output slot
- Returns money
Action: authentication

Parameters

Internal functionality

Returns
Action: authentication

Parameters
- Identity (card in slot)
- Institution (card in slot)
- PIN (typed on keypad)

Internally, it contacts institution and matches against its database, institution noted for all subsequent actions (example of state)

Returns
- Screen message ("Invalid PIN" or menu of available actions)
Action: specify_account

Parameters
Internal functionality
Returns
Action: specify_account

Parameters
- Account (touch screen from menu of choices)

Internally, choice noted for all subsequent actions (another example of state)

Returns
- None
Action: amount

Parameters

- Dollars_and_cents (typed on keypad)

Internally, amount noted (another example of state)

Returns

- Success or failure (state dependent, for example for a withdraw failure when dollars_and_cents exceeds balance)
Protocol: cash_withdrawal

What is the sequence of actions?
Protocol: cash_withdrawal

- **authentication**: failure
- **choose objective**: other objectives
- **account**: no accounts
- **amount**: balance exceeded!