Class Announcements

- Midterm Tuesday 2/10
- 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
Three-tier client/server

Client

Application server

Enterprise data server
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

- Physical
  - Bits
  - Frames
  - Packets
  - Messages
- Application
- Transport
- Network
Software Layering

- Application
- Middleware
- Operating System
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:

Infrastructure

Packet → Packet

Internet

Packet → Packet

Server

Application

<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;}
!--> ...
</html>
3-Tier Client Server Architecture example

Client

Clicks, keystrokes

Application Server

What is Bob’s balance?

$0.50

Shared data

Client

Balance $0.50

What is Bob’s balance?

$0.50

Shared data

Client

Balance $0.50

What is Bob’s balance?

$0.50

Shared data
3-Tier Client Server Architecture example
3-Tier Client Server Architecture example

Client

Application Server

Web Server

Application Logic

What is Bob’s Balance?

Database Management System (DBMS)

Database

Shared data
3-Tier Client Server Architecture example

Client

Application Server

Web Server

Application Logic

Database Management System (DBMS)

Database

What is Bob’s Balance?

Shared data

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

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

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

- **Shared data**
  - Support multiple applications with common data
  - Protect critical data
  - Decouple data administration and application administration
A diagram illustrating the interaction between different entities such as customers, financial institutions, book distribution centers, consumer, enterprise, and inter-enterprise. The diagram includes a globe with recycling symbol, representing networked applications. The image is adapted from slides for "Understanding Networked Applications" by David G. Messerschmitt. Copyright 2000. See copyright notice.
Architecture

- 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

HEADQUARTERS

Airline Dataserver

Wireless Link

Seat back devices

Wireless Link

servers
When a module is composed of sub-modules, the architecture is hierarchical.
HHC Architecture

- Seatback Application
  - Linux OS
  - Networking Infrastructure
- Coordination With On Plane Server
- User Interface
  - Data Management

- 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

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

- HHC Application
- Linux
- Networking Infrastructure
- Communication with HHC
- Communication with airline database
Interfaces

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.

```
Computation of key statistics
```

```
Compute Mean and Variance
```

```
N numbers of Float type
```

```
2 Numbers of float type that signify: Mean, Variance
```
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

PARAMETERS

N numbers of Float type

INTERFACE

Compute Mean and Variance

2 Numbers of float type that signify: Mean, Variance

RETURNS

Computation of key statistics
Implementation

Module A

Computation of key statistics

Module B

Compute Mean and Variance

\[ X_i, i=1..N \]

\[ \text{MEAN, VARIANCE} \]

Implementation 1:

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

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

- 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
Implementation

Computation of key statistics

Module A

$X_{i=1..N}$

Module B

Compute Mean and 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

Module A

Computation of key statistics

Module B

Compute Mean and Variance

Implementation 1:

\[
X_i, i=1..N
\]

\[
\text{MEAN, VARIANCE}
\]

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

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

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

- Should he use it?
  - NO!!!! Why??

- Either A should compute “SUM” himself, or sit down with B and redesign the
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 module's implementation details inaccessible to other modules is called encapsulation.
Interfaces

This simple interface example allows for only one action of module B.
- Action is “Compute mean and variance.”

Other examples are possible.