TIM 50 - Business Information Systems

Lecture 10 & 11

Instructor: Ram Akella
UC Santa Cruz

October 27 and 29, 2015
Class Announcements

- **Midterm** Thursday 10/22
- **How did it go?**
  - Solutions discussed in class today

- **Assignment 3**
  - Out today, Tues, 10/27
  - Due back Tues, 11/3

- **Pop quiz??**
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
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
- Link
- Network
- Transport
- Application

Messages → Packets → Frames → Bits → Signals
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
<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;}
//-->
…
3-Tier Client Server Architecture example

Client

Clicks, keystrokes

Application Server

What is Bob’s balance?

$0.50

Shared data

Balance $0.50

Client

Application Server
3-Tier Client Server Architecture example

Client

Application Server

Web Server

Common Gateway Interchange

Application Logic

Shared data
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

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

What is Bob’s Balance?

Database Management System (DBMS)

Web Server

Application Server

Client

Shared data

Database
### 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
  - Support multiple applications with common data
  - Protect critical data
  - Decouple data administration and application administration

- Shared data
Book distribution centers
books4u.com
Customers
Financial institution
Book distribution centers
Consumer  Enterprise  Inter-enterprise

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

- 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

- Compute Mean and Variance
  - List of numbers
  - Mean, Variance

- Computation of key statistics
- Communication with HHC
  - HHC Application
  - Linux
  - Networking Infrastructure
  - 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.

```
N numbers of Float type
```

```
Compute Mean and Variance
```

```
Computation of key statistics
```
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**

- 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**

**Module B**

Compute Mean and Variance

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

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

- Implementation 1:
  - 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

Module B

Compute Mean and Variance

\[ X_{i}, i=1\ldots N \]

\[ \text{MEAN, VARIANCE} \]

Implementation 2:

\[
\begin{align*}
\text{SUM} &= \sum_{i=1}^{N} x_i \\
\text{MEAN} &= \frac{\text{SUM}}{N} \\
\text{VARIANCE} &= \sum_{i=1}^{N} (x_i - \text{MEAN})^2
\end{align*}
\]

- 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:

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

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

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

“\( X_i, i=1..N \) 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 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..
Protocol

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
Protocol Example

Hello: I’m the HHC of Airplane#1234
Hello: I’m the gate 32 server
These were the unruly passengers on last flight
“Passengers noted”
Tell me about the passengers of my next flight
Return Passenger Data
Tell me about the weather at my next destination.
Return Weather Data

(Might be passed As an array of a compound data type “passenger,” which in turn is composed of standard types like integer, and string)
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!
More on layering

by

David G. Messerschmitt
Goals

Understand better

- how layering is used in the infrastructure
- how it contains complexity
- how it coordinates suppliers
- how it allows new capabilities to be added incrementally
Interaction of layers

Layer above is a client of the layer below

---

Layer below as a server to the layer above

---

Each layer provides services to the layer above...

---

...by utilizing the services of the layer below and adding capability
Layering builds capability incrementally by adding to what exists.
Three types of software

Application

• Components and frameworks:

  What is in common among applications

• Infrastructure:

  Basic services (communication, storage, concurrency, presentation, etc.)
Part of Microsoft vs. DOJ dispute
Major layers

- Applications
- Application frameworks and components
- Middleware
- Operating system
- Network
Data and information

Application
Deals with information

Assumes structure and interpretation

Infrastructure
Deals with data

Ignores structure and interpretation
Data and information in layers

- The infrastructure should deal with data, or at most minimal structure and interpretation.

- The application adds additional structure and interpretation.

- This yields a separation of concerns.
Package = file, message

In the simplest case, the infrastructure deals with a package of data (non-standard terminology)

- collection of bits
- specified number and ordering

The objective of the infrastructure is to store and communicate packages while maintaining data integrity

File for storage, message for communication
Data integrity

Retain the
  - values
  - order
  - number
of bits in a package
Example 1

Bob sends a letter to Alice

Bob

Envelope

US Postal Service

Shipping Container

ABC Airlines

Alice

Envelope

UK Royal Mail

Shipping Container
Example 3

HHC Server Application

Windows OS

Networking Infrastructure (Contains: TCP/IP, WiFi)

Passenger Information

HHC Client Application

Palm OS

Networking Infrastructure (Contains: TCP/IP, WiFi)

Collection of Packets
Example 3: Network Infrastructure Expanded

- HHC Server Application
  - Windows OS
  - TCP transport layer
  - WiFi Link Layer
  - WiFi Physical Layer
  - Networking Infrastructure

- Passenger Information
  - Packets
  - Radio Signals

- HHC Client Application
  - Palm OS
  - TCP transport layer
  - WiFi Link Layer
  - WiFi Physical Layer
  - Networking Infrastructure
Example 4

HHC Server Application

Windows OS

Networking Infrastructure Layers within TCP/IP, WiFi

message

“Send me today’s flight information”

HHC Server

HEADQUARTERS

Airline Dataserver

DBMS

message

Unix OS

Networking Infrastructure Layers within: TCP/IP, WiFi

Collection of Packets
Information in the infrastructure

Sometimes it is appropriate for the infrastructure to assume structure and interpretation for data:

- to add capabilities widely useful to applications
- to help applications deal with heterogeneous platforms, where representations differ

At most, data types
Data and information

Application
Deals with information

Infrastructure
Deals with data types

Assumes standard data types

Assumes structure and interpretation