ISM 50 - Business Information Systems
Lectures 14
Instructor: John Musacchio
UC Santa Cruz
November 13, 2008
Class announcements

For Next Class
- Read: MySQL Database Case

For next Thursday
- Assignment 4

Database tutorials:
- Wed. 11/12/08, 9:30-11:00am, Jack Baskin 109;
- Fri. 11/14/08, 2:00-3:30pm, Jack Baskin 109.

Student talks Tuesday 11/18:
- Trevor Wood (MySQL)
- David Kuepfer (Amazon)
Student Presentations

Katherine Beeskau (Business paper)
Alba Beltran (Business paper)
Implementation

Module A

Computation of key statistics

Module B

Compute Mean and Variance

Implementation 1:

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

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 \textit{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
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

(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
- 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
- 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)
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

Slides modified from those by David G. Messerschmitt
Example 1

Bob sends a letter to Alice

Bob

Envelope

US Postal Service

Shipping Container

ABC Airlines

Alice

Envelope

UK Royal Mail

Shipping Container
Interaction of layers

Layer above is a client of the layer below

Each layer provides services to the layer above...

....by utilizing the services of the layer below and adding capability

Layer below as as a server to the layer above
### Three types of software

<table>
<thead>
<tr>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Components and frameworks:</td>
</tr>
<tr>
<td>What is in common among applications</td>
</tr>
<tr>
<td>• Infrastructure:</td>
</tr>
<tr>
<td>Basic services (communication, storage, concurrency, presentation, etc.)</td>
</tr>
</tbody>
</table>
Data and information

Application
Deals with information

Assumes structure and interpretation

Infrastructure
Deals with data

Ignores structure and interpretation
Architecture

HEADQUARTERS

Airline Dataserver

Airline Intranet

HHC Server

Wireless Link

HHC

Airline Dataserver
Two ways to design a system

Decomposition from system requirements

Assembly from available components

Available components

System requirements

Requirements

Slide adapted from slides for *Understanding Networked Applications* by David G Messerschmitt. Copyright 2000. See copyright notice.
Components

**Component**: A subsystem purchased “as is” from an outside vendor

(Accessory – building your own subsystem)

A component implementation is encapsulated (although often configurable)

Slide adapted from slides for *Understanding Networked Applications*
By David G Messerschmitt. Copyright 2000. See copyright notice.
The Palm OS we are buying “off the shelf” and integrating into our architecture. The Palm OS is a component.
Other Examples of components

- Computer
- Disk drive
- Network
- Network router
- Operating system
- Integrated circuit
- Database management system

Why is a component implementation encapsulated?
Interoperability

Components are interoperable when they interact properly to achieve some desired functionality

Increasingly component interoperability cannot be dependent on end-user integration

- PC and peripherals
- Enterprise, inter-enterprise, consumer applications
- Role for standardization
Outsourcing: A subsystem design is contracted to an outside vendor.

Responsibility is delegated.

Slide adapted from slides for Understanding Networked Applications by David G. Messerschmitt. Copyright 2000. See copyright notice.
Suppose we choose to pay another firm to develop the user interface.

This is called **Outsourcing**.

Why would we do this?
Suppose we bring together all these subsystems and test them...

This is called **System Integration**
**System integration**

- Bring together subsystems;
- make them work together;
- to achieve a goal.

**Requires**
- Testing
- Making modifications to
  - architecture and/or
  - subsystem implementation

Slide adapted from slides for *Understanding Networked Applications* by David G Messerschmitt. Copyright 2000. See copyright notice.
Can System Integration be Outsourced?

- Of course!
Supplier Types

- Three types of suppliers:
  - Component Suppliers
  - Custom Subsystem Developers
  - System Integrators
- (Some suppliers are 2 or even 3 of above.)
Two ways to sell Software

**Product**
- Customer installed and operated
- Often (but not necessarily) sold or licensed at a fixed price

**Service**
- Functionality provided over a wide-area network
- Often (but not necessarily) sold by subscription

Slide adapted from slides for *Understanding Networked Applications* by David G Messerschmitt. Copyright 2000. See copyright notice.
Recall: Infrastructure and Applications

Infrastructure
- Equipment and/or software used by many applications

Applications
- Provide specific capabilities and features serving individual users.
Four possibilities

<table>
<thead>
<tr>
<th>Product</th>
<th>Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft Office</td>
<td>Hotmail</td>
</tr>
<tr>
<td>Application</td>
<td>Internet DNS</td>
</tr>
<tr>
<td>Infrastructure</td>
<td></td>
</tr>
<tr>
<td>Personal computer</td>
<td></td>
</tr>
</tbody>
</table>

Slide adapted from slides for *Understanding Networked Applications* by David G Messerschmitt. Copyright 2000. See copyright notice.
Application Service Provider

Two types

- Bundled
  - An infrastructure provider bundles applications with their infrastructure
    - Example: AOL, telephony service providers

- Unbundled
  - A provider of an application service without providing an infrastructure service
    - Examples?
Examples of unbundled ASP model

- Yahoo: Web-based calendar
- Hotmail: Web-based email
- Schwab: Web-based stock trading
Unbundled ASP model

Advantageous to user

- Proven way to reduce installation, integration, and maintenance costs
- Contractual obligation for availability and quality
- Location independence

Slide adapted from slides for Understanding Networked Applications
By David G Messerschmitt. Copyright 2000. See copyright notice
Unbundled ASP model (con’t)

Advantages to supplier

- Ongoing revenue stream supporting upgrade and maintenance
- Usage-based revenue better aligned with user’s value proposition
- Opportunity for price discrimination, advertising revenue, etc.

Slide adapted from slides for Understanding Networked Applications
By David G Messerschmitt. Copyright 2000. See copyright notice
Some pricing alternatives

Price discrimination?
Usage dependent?
Terms and conditions
- fixed, leasing, per-use, subscription
- warrantee, service level agreements

Bundles
- maintenance, support, releases, provisioning and operations

Who pays?
- sometimes not the end user

Slide adapted from slides for *Understanding Networked Applications*
By David G Messerschmitt. Copyright 2000. See copyright notice
Infrastructure acquisition

Infrastructure

- Build and operate
- Build but do not operate
- Do not build but operate
- Neither

Trend

Outsourced operations
System integrator
Service provider

Slide adapted from slides for Understanding Networked Applications By David G Messerschmitt. Copyright 2000. See copyright notice
Application acquisition

Application

Develop internally

Buy as product

Contract development

Product w/ customization

Trend

Software supplier

Outsource developer

Supplier, consultants