Class announcements

- For Next Class
  - Read: MySQL Database Case

- For next Thursday
  - Database Project Due

- Database tutorials
Student Presentations
BUSINESS PAPER
Business Analysis Paper

- The paper should answer questions like:
  - What industry does the company compete?
  - What is the competitive environment like?
    - “Porter model” (competitors, substitute products, buyers, suppliers, new entrants...)
  - How did your company use IT to gain a competitive advantage?
  - (Detailed guidelines are posted on the website)
How did the company use IT to support or enable its business processes and competitive strategies?

- What technologies in particular did it use?
- How much of the company’s success do you attribute to its use of IT and/or the company’s early adoption of IT into their business processes?
Citing Sources

- Plagiarism is illegal and cheating and will not be tolerated!!!
- More than thirty words verbatim must be cited.
- Any facts or figures that are not your own must be cited.
  - Ebay’s revenues in US Revenues in 2002 were $1.39 billion [1].

“Semiconductors have found a place in virtually every electronic device in existence. This helps explain why the industry was able to reach $200 billion in sales before a slump brought the figure back down in 2001” [1].

End Note:

Citing Sources

- The easiest way to lose points on your paper is to not cite sources!

- Guide on class website is posted to help you cite your sources correctly.

- Talk to the TA or instructor if you have questions.
Suggested sources of Information

- Company website
- 10K report
  - (This is the annual report public companies file with Security and Exchange Commission.)

- Business article database
  - A database of articles from magazines like “Business Week” and economics journals.
  - Find it at: [http://library.ucsc.edu](http://library.ucsc.edu)
  - Click on “articles databases” on left margin.
  - Click on “By subject”
  - Click on “economics”
  - Try this tonight! And let us know if you have problems

- Industry specific publications
  - Example: Aviation Week

- Books
- Good Magazines (*The Economist*)
- Consulting groups: Forrester, Gartner, ...
More on layering

Slides modified from those by David G. Messerschmitt
Data and information

Application
Deals with information

Assumes structure and interpretation

Assumes standard data types

Infrastructure
Deals with data types
Architecture

HEADQUARTERS

Airline Dataserver

Airline Intranet

HHC Server

Wireless Link

HHC
Two ways to design a system

- Decomposition from system requirements
- Assembly from available components

Available components

Slide adapted from slides for Understanding Networked Applications
By David G Messerschmitt. Copyright 2000. See copyright notice
Component: A subsystem purchased “as is” from an outside vendor

(Alternative – 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?

Slide adapted from slides for Understanding Networked Applications
By David G Messerschmitt. Copyright 2000. See copyright notice
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
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.
Industry Structure (cont’d)
Four possibilities

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

Infrastructure

<table>
<thead>
<tr>
<th>Build and operate</th>
<th>Build but do not operate</th>
<th>Do not build but operate</th>
<th>Neither</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outsourced operations</td>
<td>System integrator</td>
<td>Service provider</td>
<td>Trend</td>
</tr>
</tbody>
</table>

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
Stovepipe vs. Integrated Infrastructure

**Stovepipe architecture**

--- or ---

**Turnkey Solution**

- Single supplier provides all encompassing solution
- (complete with infrastructure)

**Integrated Infrastructure**

- Separate infrastructure that can support many applications

Slide adapted from slides for *Understanding Networked Applications*

By David G Messerschmitt. Copyright 2000. See copyright notice.
From stovepipe to layering

Many applications

Integrated Infrastructure
(Maybe broken into Additional layers.)

Application-independent

Application-dependent infrastructure
Stovepipe vs. Integrated Infrastructure

- What are some examples of each?
- What are the advantages of each approach?
Vertical Integration vs. Diversification

- A company is **vertically integrated** when it makes rather than buys the subsystems in its products.

- A **diversified** company produces products across different industry segments.
Vertical Integration vs. Diversification

Why do customers favor less vertical integration?

- Prefer competition amongst component suppliers
- Mix and match components
- Reduced lock in

Disadvantages??

- Customer needs to integrate components from different suppliers.

Slide adapted from slides for Understanding Networked Applications
By David G Messerschmitt. Copyright 2000. See copyright notice
Vertical Integration vs. Diversification

Why do customers favor diversification?
- Reduce coordination costs by having to deal with fewer suppliers.
General Trend

- Less Vertical Integration
- More Diversification
- Of course there are exceptions...
Today's supplier structure

Applications

Frameworks and components

Middleware

Infrastructure (network, OS) software

Equipment (network, computers)

Semiconductors, components

Slide adapted from slides for *Understanding Networked Applications*
By David G Messerschmitt. Copyright 2000. See copyright notice
Role of Venture Capital in Computing.

- Open interfaces allow small firms to contribute components without having to develop entire solution.

- Fast decision making and no supplier lock-in.

- Other Advantages?
Standardization
Purpose of a standard?

- Allow products or services from different suppliers or providers to be interoperable.
Scope of a standard

Included:
- interfaces (physical, electrical, information)
- architecture (reference model)
- formats and protocols (FAP)
- compliance tests (or process)

Excluded:
- implementation
- (possibly) extensions
Reference model

Decide decomposition of system
- where interfaces fall

Defines the boundaries of competition and ultimately industrial organization
- competition on the same side of an interface
- complementary suppliers on different sides
- hierarchical decomposition at the option of suppliers
- (possibly) optional extensions at option of suppliers

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

Once a standard is set

- becomes possible source of industry lock-in; overcoming that standard requires a major (~10x?) advance
- may lock out some innovation

In recognition, some standards evolve

- IETF, CCITT (modems), MPEG
- backward compatibility
Types of standards

*de jure*
- Sanctioned and actively promoted by some organization with jurisdiction, or by government

*de facto*
- Dominant solution arising out of the market
- Voluntary industry standards body

Industry consortium
Common or best practice

Examples?
Examples

*de jure*
- GSM, ISDN Telephone interface

*de facto*
- Microsoft Windows API (Application Programming Interface)
- Intel Pentium instruction set,

Voluntary industry standards body
- IEEE (Institute of Electrical and Electronic Engineers)
- IETF (Internet Engineering Task Force)

Industry consortium
- W3C (World Wide Web Consortium)
- SET (Secure Electronic Transactions)

Best practice
- Windowed GUI
The changing process

- As technology and industry move more quickly, the global consensus standards activity has proven too unwieldy
  - e.g. ISO

- “New age” standards activities are more informal, less consensus driven, a little less political, more strategic, smaller groups
  - e.g. OMG, IETF, ATM Forum, WAP

Programmable/extensible approaches for flexibility
  - e.g. XML, Java
Old giving way to the new
Reasons for change

- From government sanction/ownership to market forces
  - Increasing fragmentation
  - Importance of time to market

Greater complexity
- Less physical/performance constraint for either hardware or software
Lock-in

(Particularly open) standards reduce consumer lock-in

- Consumers can mix and match complementary products

Increase supplier lock-in

- Innovation limited by backward compatibility
- e.g. IP/TCP, x86, Hayes command set
Aside: Network Effects

- The value of owning some products goes up if lots of other people have it too.
  - Examples?

- This phenomenon is called “network effects”

- How do standards influence network effects?
Network effects

Standards can harness network effects to the industry advantage

- Revenue = (market size) \times (market share)

Increases value to customer

Increases competition

- Only within confines of the standard
- But forces customer integration or services of a system integrator
Why standards?

*de jure* are customer driven to reduce confusion and cost
*de facto* standards are sometimes the result of positive feedback in network effects

Customers and suppliers like them because they
- increase value
- reduce lockin

Governments like them because they
- promote competition in some circumstances
- May believe they can be used to national advantage
Approaches

Consensus
- ISO

Collaborative design
- MPEG

Competitive “bake off”
- IETF

Coordination of vendors
- OMG

Slide adapted from slides for *Understanding Networked Applications*
By David G Messerschmitt. Copyright 2000. See copyright notice
Open vs. Proprietary Standards

- Open standard - a standard that is well documented, unencumbered by intellectual property rights and restrictions, and available to any vendor.

- What are the advantages?

- What are the disadvantages?
Why companies participate

Pool expertise in collaborative design
- e.g. MPEG

Have influence on the standard

Get technology into the standard
- Proprietary, with expectation of royalties
- Non-proprietary

Reduced time to market
Standards applied to Business Processes?

- Can you standardize business processes?

- Yes!
  - **ISO 9000**
    - A set of standardized business processes for Quality Management.
    - Supports TQM (Total Quality Management)
  - **RosettaNet**
    - A set of standardized business processes, and accompanying standardized data interfaces/formats for conducting e-business.
Databases

by

David G. Messerschmitt
Databases

Treat data as a separate asset
- May be shared by multiple applications

Provide protection and integrity features appropriate to mission-critical data
- Access control
- Integrity constraints
- Persistence
- etc.
Two capabilities

Application I

**Aggregation**: accessing multiple databases

Application II

**Sharing**: two or more applications accessing the same databases

Databases
## Relational table

A relational table can be represented as a table with columns and rows. Here is an example:

<table>
<thead>
<tr>
<th>Table</th>
<th>Employee</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Address</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Record</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Table**: The overall structure or framework.
- **Employee**: A specific row within the table.
- **Name**: A column within the Employee row.
- **Address**: Another column within the Employee row.
- **Dept**: A third column within the Employee row.
- **Record**: The entire Employee row as a record.
- **Field**: Any specific item within a column (e.g., Name, Address, Dept).
SQL interface

- SQL (Structured Query Language)
- Presents single abstract interface to the application logic
  - For manipulating, and extracting data from database
- Standardized, not vendor specific

- Encapsulates various internal details
  - Data partitioning and replication
  - Host mapping
  - File representation
  - etc.
Database operations

Each operation results in a new table
Multiple tables

“PROJECT”

“SELECT”
# Database Operations

## Passengers

<table>
<thead>
<tr>
<th>Name</th>
<th>Dept ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>1</td>
</tr>
<tr>
<td>Bob</td>
<td>1</td>
</tr>
<tr>
<td>Chris</td>
<td>2</td>
</tr>
</tbody>
</table>

## Departments

<table>
<thead>
<tr>
<th>Dept Name</th>
<th>Dept ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Engineering</td>
<td>1</td>
</tr>
<tr>
<td>Sales</td>
<td>2</td>
</tr>
</tbody>
</table>

JOIN

<table>
<thead>
<tr>
<th>Name</th>
<th>Dept ID</th>
<th>Dept Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>1</td>
<td>Engineering</td>
</tr>
<tr>
<td>Bob</td>
<td>1</td>
<td>Engineering</td>
</tr>
<tr>
<td>Chris</td>
<td>2</td>
<td>Sales</td>
</tr>
<tr>
<td>Year</td>
<td>City</td>
<td>Accommodation</td>
</tr>
<tr>
<td>------</td>
<td>---------</td>
<td>-------------------</td>
</tr>
<tr>
<td>2002</td>
<td>Oakley</td>
<td>Bed&amp;Breakfast</td>
</tr>
<tr>
<td>2002</td>
<td>Oakley</td>
<td>Resort</td>
</tr>
<tr>
<td>2002</td>
<td>Oakland</td>
<td>Bed&amp;Breakfast</td>
</tr>
<tr>
<td>2002</td>
<td>Oakland</td>
<td>Resort</td>
</tr>
<tr>
<td>2002</td>
<td>Berkeley</td>
<td>Camping</td>
</tr>
<tr>
<td>2002</td>
<td>Berkeley</td>
<td>Bed&amp;Breakfast</td>
</tr>
<tr>
<td>2002</td>
<td>Berkeley</td>
<td>Resort</td>
</tr>
<tr>
<td>2002</td>
<td>Albany</td>
<td>Camping</td>
</tr>
<tr>
<td>2002</td>
<td>Albany</td>
<td>Bed&amp;Breakfast</td>
</tr>
<tr>
<td>2003</td>
<td>Oakley</td>
<td>Bed&amp;Breakfast</td>
</tr>
<tr>
<td>2003</td>
<td>Oakley</td>
<td>Resort</td>
</tr>
<tr>
<td>2003</td>
<td>Oakland</td>
<td>Bed&amp;Breakfast</td>
</tr>
<tr>
<td>2003</td>
<td>Oakland</td>
<td>Resort</td>
</tr>
<tr>
<td>2003</td>
<td>Berkeley</td>
<td>Camping</td>
</tr>
<tr>
<td>2003</td>
<td>Berkeley</td>
<td>Bed&amp;Breakfast</td>
</tr>
<tr>
<td>2003</td>
<td>Berkeley</td>
<td>Resort</td>
</tr>
<tr>
<td>2003</td>
<td>Albany</td>
<td>Camping</td>
</tr>
<tr>
<td>2003</td>
<td>Albany</td>
<td>Bed&amp;Breakfast</td>
</tr>
</tbody>
</table>

- Entries are simple data types or compositions of those types
  - Integer, string, etc.
Object-relational database

- A column can store object instances of a given class rather than data of a given simple or compound data type
- Because of the table structure, SQL can be extended to this case
- Standard SQL queries can be extended to methods returning simple data types
- Many other good ideas
Benefits of ORDBMS

Extension: manage arbitrarily complex data types
Migration: preserve and extend existing databases
Preserve SQL interface
  - OR extensions in latest standard

All the benefits/experience of earlier databases
  - Access control, data integrity, persistence, etc.

Killer app: Behind Web/CGI
  - Images, video, audio, animation, applets, etc.
Algorithms and protocols

Adapted from
David G. Messerschmitt
Algorithm

Specified sequence of steps that
- accomplish a designated task
- in a finite number of steps

Representation:
- simple algorithm: flowchart
- complicated algorithm: program
Example: one turn at monopoly

Start turn → Throw dice → Move token number of squares indicated on dice

Land on “go to jail”? → Yes → Move to “jail” square → Finish turn

Land on “go to jail”? → No → Do not move; follow policies for square (like “pay rent”) → Finish turn
Algorithm building blocks

Programming languages support these three building blocks:

- **Sequence**
  - Start
  - Action
  - Action
  - Action
  - Finish

- **Selection**
  - Start
  - Decision
  - Action
  - Action
  - Finish

- **Loop**
  - Start
  - Test
  - Action
  - Finish
Protocol

- Distributed algorithm ...
- Realized by two or more modules to coordinate their actions or accomplish some shared task

Module interoperability requires a protocol

- Prescribed order of method invocations
- Part of interface documentation
Monopoly players protocol

This is a protocol interaction diagram

Player 1

Player 2

One-turn algorithm

Time
Internally, the network uses protocols to implement the services it provides.

The application defines its own application-level protocols.
Example:

HEADQUARTERS

Airline Dataserver

HHC Server

Wireless Link

HHC

Airline Intranet
Layered Protocols Example

HHC Server

HHC Server Application

Windows OS
Break Messages into Packets
Networking Infrastructure

Request Pass. Data

Application Level Protocol

Send Pass. Data As Message

HHC

HHC Application

Palm OS

Networking Infrastructure

Send Packet

Acknowledge Packet

Link Level Protocol
Three simple protocols

One-way message: send-receive
Two-way interaction: request-response
Push: publish-subscribe
Send - Receive
Request - Response

Client

request

response

Server
Send - Acknowledge

Client

Server

send

acknowledge
Example: HTTP (Hyper Text Transfer Protocol)

1. User activates URL
2. HTTP request
3. HTTP response (embedded document)
4. Browser displays document (if HTML) or invokes “helper application”

HTTP client (browser)

HTTP server

HTML documents
Locating things

by

David G. Messerschmitt
Three ways of locating things

Name

- “Joe Bloe”

Address

- “1299 Hearst St, Berkeley, CA”

Reference

- “Postmaster of Berkeley CA”
Name

- Symbolic (character string) representation
- Easy for people to remember or guess
- Identifies, but
  - Does not locate directly
    - Distinction important for mobile entities
- Not unique: entities can have more than one name (called aliases)
Hierarchical names

Hierarchy makes names easier to remember or guess

Host domain names:

- “info.sims.berkeley.edu”
- designates administrative hierarchy

File names:

- “c:\My Documents\Docs\Resume.doc”
- designates folder hierarchy
Address

- Route or path to entity
  - is directly specified, or
  - can be inferred

- Independent of who or what is locating entity

- Topological specification
Example

Path from \[\text{cyan}\] to \[\text{olive}\] is \((R,D,D,D,R,R,R,R)\)

Is \((R,D,D,D,R,R,R,R)\) an address?
No! -- not an address, because it depends on starting point
Example

Address of \( (6,5) \) is (6,5)

Route from \( (6,5) \) can be inferred
Abstract representation of an entity

Interaction is with representation

- infrastructure arranges redirection to actual entity
- especially appropriate for things that move

Example

- A Cell phone number is a reference.
- A Wired phone number is an address.
Name services

1. name

2. address or reference

3. interaction
Markup languages
A markup language describes the structure of a document

- Based on tags
- Tags denote structural elements like sections, subsections, figures, etc

Internationally standardized, so application independent
Example: HTML

<html>
<h1> Super Widget </h1>
<h2> Widgets Incorporated </h2>
<em> 123456789 </em>
<br>
<p> $300 </p>
</html>
Example: XML

Tags Emphasize what the things *mean* rather than how to *format* their Presentation.

```xml
<xml>
  <product>
    <model> Super Widget </model>
    <make> Widgets Incorporated </make>
    <sku> 123456789 </sku>
    <price> $300 </price>
  </product>
</xml>
```
XML in Ecommerce example

Stuff4U

Super Widget  $300
Amazing Gadget  $500

Supplier

Product info
From each Supplier sent in XML

Retailer

Consumer
XML in ecommerce example 2

<xml>
  <product>
    <model> Super Widget </model>
    <make> Widgets Incorporated </make>
    <sku> 123456789 </sku>
    <price> $300 </price>
  </product>
</xml>

Product info
From each Supplier sent in XML

XYZ Manufacturing

Super widget recognized and managed by SCM software.
Family lineage

SGML
- Standardized in mid 80s by ISO

HTML
- Introduced in Early 90s
- Emphasizes formatting and presentation of documents

XML
- Proposed in mid 90s
- Emphasizes structure of documents
- Purpose- and industry-specific extensions
### Distribution for Midterm

**Statistics: Midterm**
- Graded out of: 100.00
- Number of records: 127
- Highest grade: 99.00
- Lowest grade: 16.00
- Mean grade: 70.58
- Median grade: 74.00
- Standard deviation: 17.27

<table>
<thead>
<tr>
<th>Score Range</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ 0, 10 )</td>
<td>1</td>
</tr>
<tr>
<td>[ 10, 20 )</td>
<td>1</td>
</tr>
<tr>
<td>[ 20, 30 )</td>
<td>1</td>
</tr>
<tr>
<td>[ 30, 40 )</td>
<td>4</td>
</tr>
<tr>
<td>[ 40, 50 )</td>
<td>12</td>
</tr>
<tr>
<td>[ 50, 60 )</td>
<td>15</td>
</tr>
<tr>
<td>[ 60, 70 )</td>
<td>18</td>
</tr>
<tr>
<td>[ 70, 80 )</td>
<td>31</td>
</tr>
<tr>
<td>[ 80, 90 )</td>
<td>29</td>
</tr>
<tr>
<td>[ 90, 100 )</td>
<td>16</td>
</tr>
<tr>
<td>[ 100 ]</td>
<td></td>
</tr>
</tbody>
</table>