Announcements

- Assignment 4 due today
- Business Paper Draft Due today
- Database Assignment 2 due 3/9
Midterm

- Average 95.2

Midterm distribution
**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

Data
Voice
Video

Many applications

Integrated Infrastructure
(Maybe broken into Additional layers.)

Application-dependent infrastructure

Application-independent
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
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
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, MPEG
- backward compatibility

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

de facto

- Microsoft Windows API (Application Programming Interface)
- Intel instruction set,

Voluntary industry standards body

- IEEE (Institute of Electrical and Electronic Engineers)
- IETF (Internet Engineering Task Force)

Industry consortium

- blu-ray

Best practice

- Windowed GUI
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. IETF

Programmable/extensible approaches for flexibility

- e.g. XML, Java
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. IPv4/TCP,
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) x (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 lock-in

Governments like them because they

- promote competition in some circumstances
- May believe they can be used to national advantage

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

Consensus
- ISO

Collaborative design
- MPEG

Competitive “bake off”
- IETF
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.
Chapter 9

Applications and the Organization
**Build vs. Buy?**

**Purchase off the Shelf**
- less time and cost
- benefits of using a “standard” solution
- support available
- must mold org to app
- no potential for competitive advantage

**Outsource**
- developers not as familiar with org as you
- more opportunity for customizing than off the shelf
- contractor may share knowledge with competitors
- contractor may have too much bargaining power

**Make**
- most customizable of 3
- easier iteration between conceptualization and development needed
- most risky
- org may lack competency to do it
Application Lifecycle

It is important to think beyond acquiring an application

- How do we come with the idea?
- How do we architect it?
- How do we implement?
- How do we extend and maintain it?

For this reason, the software engineering community came up with:

- Application Lifecycle Model
Application Lifecycle

Stages:
1. Conceptualization
2. Analysis
3. Architecture Design
4. Development Evolution
5. Testing and Evaluation
6. Deployment
7. Operations, Maintenance, and Upgrade
1) Conceptualization

What is the vision?
- What are the objectives?
- What is the business case?

- EXAMPLE: Seatback system to sell seat swaps

- Business Case:
  - Increase revenue, passenger satisfaction
Conceptualization

- New in-flight seatback system
  - Sell upgrades and seat swaps
    - (People who want to get away from sick people ...)
  - Offer to exchange seats
Invent a concept for a new service that depends in large part on IT.

- (can be a service that enhances offerings of an existing company)

Examples:

- Service that automatically reroutes incoming UPS packages to a person when they are traveling.
- Service that automatically rebooks a person on alternative flights when there are delays or cancellations.
- Service to give or receive shares of cooked meals between neighbors

Write your idea down.
2) Analysis

- Describe what the application will do.
- Enough info to allow “stakeholders” to review idea
- Don’t make highly detailed specifications
- Describe scenarios in which it is used
  - (Use Cases)
2) Analysis -- Example

- **Example Use Case**

- **Seat Trade**
  - Passenger in 10C (aisle) indicates willingness to trade seat for any other seat on the plane plus $100.

  - Passenger 28B (Middle) indicates willingness to pay $200 to trade seat for an aisle.

  - Matching algorithm arranges a swap between 10C and 28B. Charges passenger originally in 28B $200, pays passenger originally in 10C $100. Airline keeps $100.
Develop a use case for your idea

- Give a specific example like in the previous slide.
- Write your example down.
3) Architecture Design

- Decompose the application into subsystems
  - Hardware, software
  - Try use commercial off the shelf subsystems
  - Try to use standard infrastructure layers
    - Operating system, network, middleware, etc.
Architecture

Wireless Link

Seat back devices

Wireless Link

servers

HEADQUARTERS

Airline Dataserver
When a module is composed of sub-modules, the architecture is **hierarchical**.
Define the functionality, interaction and interfaces of subsystems

While doing this, consider

- **Scalability**
  - How easily can we increase the number of users and maintain performance?

- **Extensibility**
  - How easily can we add new features in the future?

- **Administration**
  - How much work will it take by humans to keep this running properly?
4) Development Evolution

- **Develop the details**
  - Develop/program custom subsystems
  - Have contractor build outsourced pieces
  - Put together with off-the-shelf components

- **Incremental**
  - Start with simplest implementation and get it working
  - Later add more features.
5) Testing

- A must!

- If architected well, we can test subsystems independently.

- Alpha test – offline test of prototype

- Beta test – test in intended environment with cooperative users
  - Example - give HHC to initial group of FA’s
6) Deployment

- Convert from previous processes if necessary
  - Example: CISCO ERP (all at once)
  - Or, you could do incrementally

- Train users

- Data importation
  - (if necessary)
7) Operations, Maintenance, Upgrade

- Maintain Security
- Repair Problems
- Correct performance short comings (Cisco ERP)
- Add features
ALM rarely followed precisely

Many times projects loop between stages

ALM followed more closely in larger companies

Alternative:
- Rapid Iterative Prototyping
  - (Cisco did some of this in the ERP case.)