Architecture

- How do you begin to architect a solution for a problem like this?

- Break it into modules!
Properties of Modularity

- **Functionality**: chosen as distinct and natural functional groupings, e.g., executive, legislative, etc.

- **Hierarchy**: itself be a system, internally decomposed into modules, not visible externally, e.g., agencies, departments

- **Separation of concerns**: strongly associated within each module, weakly to other modules, e.g., law, legislative

- **Interoperability**: successfully interact to realize higher level purposes of the system, e.g., law enforcement, judiciary

- **Reusability**: defined, documented, implemented independently so their designs can be used by other systems, e.g., US government, adopted by other countries
Granularity tradeoff.

- How big should we make the modules
  - Many simple small ones (fine: divide and conquer)
  - Or a few complicated big ones... (easy to understand)

- This aspect of modularity is called \textit{granularity}.

- Which is better? Typically “decomposition within modules” allows the system to view different level of granularity
Simple modularity in a computer system

- Arithmetic unit
- Address unit
- Processor
- Memory
- Storage
- Network adaptor
- Network
- Bus
Simple modularity in a computer system

- Processor
  - Arithmetic unit
  - Address unit
- Memory
- Storage
- Network adaptor
- Network
- Bus

interface
Interface

- An external view of a module, specifying what functions it performs and instructing other modules to invoke those functions.
- Determined at time of architecture was designed.
- Could be with different versions of a module share a common interface or module changes over time.
  - leads to “separation of concerns”, e.g., equipment module: printers (not specify how printer does its job).
Interface

- Specified at time of the architecture is designed without assumption of how the module will be implemented.
- Serve as a starting point when the implementation of a module is turnovered to other people.
- Hardware interface: power outlets, USB.
- Software interface: boundary between two software programs, i.e., think each software is a server, “interface” provides way that two servers can communicate.
Interfaces: three properties of data passed through an interface

Compute Mean and Variance

PARAMETERS

N numbers of Float type

INTERFACE

Computation of key statistics

RETURNS

2 Numbers of float type that signify: Mean, Variance

Name, Type, and Value
Data: Type, Name and Value

- **Name**: meaningful to users, e.g., “dailyrain”, “income”

- **Type**: structure & interpretations, passing an interface is often specified in terms of a limited number of standard data types,
  - e.g., integer, float (m,n), characters, and string (range of values and allowable manipulation)

- **Value**: changes over time due to manipulations
Example data types

Integer
- “natural number between, limited in size by the number of bits”
- Could be represented (in many ways) by 16 bits, e.g., since $2^n = 65,536$ (n=16)
  - E.g., The number of letters in the Roman alphabet is 26.

Float
- Real number approximated by a finite number of digits in scientific notation.
- In form of $m \times 10^n$, where size of $n$ and $m$ is limited by the number of bits
More data types

Character

- “values assuming a-z and A-Z plus space and punctuation marks”
  - could be represented by 8 bits (ASCII) or Unicode (16 bits)

Character string

- “collection of $n$ characters, where $n$ is customizable”
Compound data types

Programmer-defined composition of basic data types

Example:

```java
Employee {
    String name;
    String address;
    Integer year_of_birth;
    etc.
}
```
**Actions & Protocols Parameters and Return**

- **Action**: something specifics that a module does
- **Protocol**: A sequence of “actions” leads to an protocol
  - Typically a menu that tells clients what it prepares to do, e.g., withdraw, deposit, balance check, transfer or +, -, ×, ÷
  - A pull-down or display menu
  - protocol “+”: 1) enter: “-1” & “-2” → display 2) add “-1+ (-2)” → display (-3)
  - “-1”, “-2” are parameters and “-3” is return.
- **Message**: send a packet of data w/o return
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
Client-Server Interactions

1) Decides it needs to invoke an action of a server module
2) Invokes the action by name
3) Passes data to server
4) Processes parameters in accordance with the specified action; generates return values
5) Passes the return values to the client
6) Processes the return values to complete the interaction

Client Module

Server Module
Human Interfaces

- Interface applies to interaction between a user and a computer application

- Graphic user interface (GUI): use comfortable elements to human
  - E.g., graphics, images, animation, sound, pointing devices, keyboards
  - Pull down menus
Interface Example:

Automatic teller machine (ATM)

What is the interface between this machine and the customer?
Steps

Define available actions
  e.g., Authentication, specify_account, amount, cash_withdrawal, and etc.

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!
Well-principled architecture

Goal:

✓ Minimize cost & maximize performance
✓ Minimize effort to maintain & develop new applications
✓ Provide capability to support operations and reliability

Rely on layering principle:

✓ To contain complexity
✓ To coordinate suppliers of modules
✓ To allow new capabilities to be added incrementally
Layering builds capability incrementally by adding to what exists.

Layering

Elaboration or specialization

Existing layers
Layering builds capability incrementally by adding to what exists.

Layering builds capability incrementally by adding to what exists.

Assembly (internal)

Inventory

Receiving

supply (external)
Interaction of layers

“Here is the actions I can perform to support you, but don’t ask how I do them”

Layer above is a client of the layer below

Each layer provides services to the layer above...w/ revealing it is done

....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>
Major layers

Network Computing Infrastructure

Applications

Application components

Middleware

Operating system

Network

Those are specialized services not typically provided within the company

Analogy

Auto manufacture and devises process of assembly lines

All auto companies components, e.g., batteries

Professional services, e.g., accounting, law

Resource management, e.g., janitorial, gardening

Support interaction at different locations
Summary of Major layers Principles

1) Each layer is a server to the layer above, providing a set of actions but not revealing how they are implemented, e.g., a new networking technology can be introduced without modifying middleware or application.

2) Each layer is a client to the layer below, utilizing its available actions in the course of providing services to the layer above.

3) Each layer is permitted to interact with the layers immediately above and below.

Functionally, provide increasingly elaborated or specialized services at each higher layer, based on services provided by lower layers!!
Infrastructure needs to “treat” each host “identically” in a sense that it needs to worry less about “types” or “heterogeneity” of applications (e.g., Windows words, Mac words or other text compatible editors).

This is done by having “middleware” that shields application from the rest of the infrastructure!
Layers on Data and Information

Applications

Middleware

Operating system

Network

Additional interpretation

Additional structure, e.g. files (storage) or message (communications)

Packets of bits, with 1) known order 2) unknown structure & interpretation

No interpretation attached until application → Separation of concerns
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 2

Application

Web server

Operating system

File system

(retrieve by browser)

File

Message

Network

Fragmentation

Collection of packets

Assembly

Web page

Web browser

Screen

HTML
**Major layers**

Network Computing Infrastructure

- Applications
- Application components
- Middleware
- Operating system
- Network

**Analogy**

- Auto manufacture and devises process of assembly lines
- All auto companies components, e.g., batteries
- Professional services, e.g., accounting, law
- Resource management, e.g., janitorial, gardening
- Support interaction at different locations

Those are specialized services not typically provided within the company.
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.
Components

When the functionality of subsystem reaches its maturity, but allows for customizing

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 Linux OS we are buying “off the shelf” and integrating into our architecture. The Linux 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
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?
System Integration

- 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
Supplier Types

Three types of suppliers:

- **Component Suppliers**
  - Specialized in one or a set of related components

- **Custom Subsystem Developers**
  - Taking customer’s requirement and meet their needs

- **System Integrators**
  - Implementing, assembling and integrating components

(E.g., Computers (microprocessor, disk, drive, etc), computer manufacture: 1 & 2

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

- Product
  - Microsoft Office
  - Personal computer

- Service
  - Hotmail
  - Internet DNS

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

**Two types**

- **Push**
  - Suppliers develop and define a product and sell it off-shelf
  - *Example:* Microsoft Windows

- **Pull**
  - Customers provide a specification and commission a supplier to develop such application
  - *Examples:* ecommon

*(Mostly a off-shelf but allows for modification!)*
Application Service Provider (ASP)

- **Two types**
  - Bundled
    - An infrastructure provider bundles applications with their infrastructure
      - Example: Comcast, telephony service providers, Ooma?
  - Unbundled
    - A provider of an application service without providing an infrastructure service
      - Examples:
Examples of unbundled ASP model

- Yahoo: Web-based calendar
- Gmail: 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
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 (e.g., standard, premium), advertising revenue, etc.
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

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 Messerschmidt. Copyright 2000. See copyright notice
Application acquisition

Application

\{ Develop internally \quad \text{Buy as product} \quad \text{Contract development} \quad \text{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

Data  Voice  Video

Application-dependent infrastructure

Many applications

Integrated Infrastructure (Maybe broken into Additional layers.)

Application-independent
Stovepipe vs. Integrated Infrastructure

**Stovepipe Architecture**
---or---

**Turnkey Solution**

- Eg., landline telephone
- One supplier involves, takes full responsibility
- E.g., vertical integrated utilities (others?)

**Integrated Infrastructure**

- Supplies focus on either applications or infrastructure.
- Easy to deploy new applications
- Layered structures

---

Application and Infrastructure

Application

Infrastructure

---

Slide adapted from slides for *Understanding Networked Applications* by David G. Messerschmitt. Copyright 2000. See copyright notice.
Stovepipe vs. Integrated Infrastructure

- **Economies of scope**: allow supporting a variety of applications
- **Economies of scale**: allow supporting a variety of applications, thereby lowering unit costs
- **Lower marginal cost**: Each new application leverages on existing infrastructure
- **Larger market**: owing to exiting infrastructure, suppliers see low risk with great potential
- **Diversity of applications**: resulted from low marginal cost & larger market
- **Competition**: User can mix and match complementary technologies.
Vertical Integration vs. Diversification

- A company is **vertically integrated** when it makes rather than buys the subsystems in its products by acquiring suppliers that previously sold it components.

- A **diversified** company produces products across different industry segments, achieve synergies, financial stability, e.g., google, costco (gas, tire), etc.
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.
Vertical Integration vs. Diversification

Why do customers favor diversification?

- Reduce internal coordination costs by having to deal with fewer suppliers or simply face a single service provider.

(because suppliers also produce products across different segments! Think about what can you do with google-related services, including google doc, calendar, photo, etc or Apple-related ones.)
General Trend

- Less Vertical Integration
- More Diversification

- More vertical integration
- Make
- Buy

- Internal coordination cost down
- External coordination cost down

- Reduce risk, product cycle
- Provide customer with complete solution
- Synergies across different product lines

- Customer demand for open system
- Competition improve quality & prices

Make Diversification
Today’s supplier structure

<table>
<thead>
<tr>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frameworks and components</td>
</tr>
<tr>
<td>Middleware</td>
</tr>
<tr>
<td>Infrastructure (network, OS) software</td>
</tr>
<tr>
<td>Equipment (network, computers)</td>
</tr>
<tr>
<td>Semiconductors, components</td>
</tr>
</tbody>
</table>

Slide adapted from slides for *Understanding Networked Applications* by David G Messerschmitt. Copyright 2000. See copyright notice.
Purpose of a standard?

- Allow products or services from different suppliers or providers to be interoperable*

- Increasingly component interoperability cannot be dependent on end-user integration, *e.g.*, think about that you have to put a computer where there are various non-standard interfaces existed.

*Components are interoperable when they interact properly to achieve some desired functionality*
Scope of a standard*

Included: *, a specification generally agreed upon

- interfaces (physical, electrical, information)
- architecture (reference model)
  - Standard way of decomposing a system so that suppliers (competitive, complementary) can follow
- formats and protocols (FAP)
  - Define how interface works, e.g., parameters, return, etc.
- compliance tests (or process)

(Establish a ongoing process of upgrading, improvement, e.g., extensions)

---

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

Decide decomposition of system

- where interfaces fall

Defines the boundaries of competition and ultimately industrial organization (think about layering decomposition)

- 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 advance, e.g., “biogas (E85)” others?
- may lock out some innovations,
  - e.g., might be alternative of doing thing that is not compatible with the standard
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? “walk on the right side”, “drive on right side”
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
- bluray

Best practice
- Windowed GUI

Slide adapted from slides for *Understanding Networked Applications*
By David G Messerschmitt. Copyright 2000. See copyright notice
The changing process

- As technology and industry progress quickly, the global consensus standards activity has proven too difficult
  - 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, i.e., not easy to regulate
- Importance of time to market, i.e., regulatory process is typically slow

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

Standard increases supplier lock-in

- Innovation limited by backward compatibility*, i.e., suppliers need to respect existing standards

(An earlier version conform and can be used in situation with new standards, e.g., MPEG.

---

Slide adapted from slides for *Understanding Networked Applications* by David G Messerschmitt. Copyright 2000. See copyright notice.
Aside: Network Effects

- The value of owning some products goes up if lots of other people have it too.
  - Examples? “dropbox”, “line”, others?

- This phenomenon is called “network effects”

- How do standards influence network effects?
Network effects

Standards can harness network effects to the industry advantage

- Revenue = price \times \text{market size} \times \text{market share}

Increases value to customer (market size)
Why standards?

*de jure* are government driven to reduce confusion and cost

*de facto* standards are sometimes the result of positive feedback in network effects, not recognized by any formal body, open used by anyone, but may be with a proprietary implementation, e.g., *post script* (ps).

Governments like them because they

- promote competition in some circumstances
- may believe they can be used to national advantage
Open vs. Proprietary Standards

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

- What are the advantages? e.g., no lock-in for consumers, stimulate innovations

- What are the disadvantages? e.g., proprietary technologies may find it difficult to compete
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

Reduced time to market

Slide adapted from slides for *Understanding Networked Applications*
By David G Messerschmitt. Copyright 2000. See copyright notice
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.
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 (closely tracking expanding requirements, lengthy)
- org may lack competency to do it
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?
  - Including 1) vision (what new function is to be accomplished?), 2) assumptions (what are fixed points that cannot be changed.)

- What is the business case?
  - Convince management that the investment is warranted.
  - Why does it make sense to do this?

- 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
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): represents a typical usage of the applications, subject to external events or actions of people or applications responding to this applications.
2) Analysis -- Example

- Example: Scenario:
  - Seat Trade
  - Passenger in 10C (aisle) offers to trade seat for frequent flyer miles
  - Business traveller in 20B (middle) offers to pay $500 to get aisle seat
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
HHC Architecture

When a module is composed of sub-modules, the architecture is **hierarchical**.
3) Architecture Continued

- 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

- Initially, a very incomplete implementation of the system, but incorporate major subsystems with details filled in.
- 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!**
- To uncover shortcomings & flaws
- Not await for a complete production, but a part of integration process.

- **If architected well, we can test subsystems independently**
  (Think about clinic trails in drug/vaccine development)

- **Alpha test - offline test of prototype**
  - Tested out in an environment that approximates the intended usage.
  - Identified major problems.

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

- Establishment of human organization and the hardware infrastructure (network, hosts) and user's training.
- Convert from previous processes if necessary
  - Example: CISCO ERP (all at once)
  - Or, you could do incrementally
- Data importation
  - (if necessary)
7) Operations, Maintenance, Upgrade

- Maintain Security
- Repair Problems
- Correct performance short comings (Cisco ERP)
- Add features
- Why continuous process
  - No amount of testing can detect and repair problems.
  - Operational requirement evolves over time.
Application Lifecycle Model concluding remarks

- 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.)
• **Database:**
  - Collection of related files containing records on people, places, or things.
  - Prior to digitizing DBs, business used paper files.

• **Entity:**
  - Generalized category representing person, place, thing on which we store info.
  - E.g., SUPPLIER, PART

• **Attributes:**
  - Specific characteristics of each entity:
    - SUPPLIER name, address
    - PART description, unit price, supplier
Relational database:

- Organize data into tables
- One table for each entity:
  - E.g., (CUSTOMER, SUPPLIER, PART, SALES)
- Fields (columns) store data representing an attribute.
- Rows store data for separate records.
- Key field: uniquely identifies each record.
- Primary key:
  - One field in each table
  - Cannot be duplicated
  - Provides unique identifier for all information in any row
A relational database organizes data in the form of two-dimensional tables. Illustrated here is a table for the entity SUPPLIER showing how it represents the entity and its attributes. Supplier_Number is the key field.

Figure 5-1
### The PART Table

<table>
<thead>
<tr>
<th>Part_Number</th>
<th>Part_Name</th>
<th>Unit_Price</th>
<th>Supplier_Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>137</td>
<td>Door latch</td>
<td>22.00</td>
<td>8259</td>
</tr>
<tr>
<td>145</td>
<td>Side mirror</td>
<td>12.00</td>
<td>8444</td>
</tr>
<tr>
<td>150</td>
<td>Door molding</td>
<td>6.00</td>
<td>8263</td>
</tr>
<tr>
<td>152</td>
<td>Door lock</td>
<td>31.00</td>
<td>8259</td>
</tr>
<tr>
<td>155</td>
<td>Compressor</td>
<td>54.00</td>
<td>8261</td>
</tr>
<tr>
<td>178</td>
<td>Door handle</td>
<td>10.00</td>
<td>8259</td>
</tr>
</tbody>
</table>

**Figure 5-2**

**Primary Key**

**Foreign Key**
Establishing relationships

- Entity-relationship diagram
  - Used to clarify table relationships in a relational database
- Relational database tables may have:
  - One-to-one relationship
  - One-to-many relationship
  - Many-to-many relationship
  - Requires creating a table (join table, Intersection relation) that links the two tables to join information
A Simple Entity-Relationship Diagram

This diagram shows the relationship between the entities SUPPLIER and PART: One-to-many

Figure 5-3
• **Normalization.** Process of streamlining complex groups of data (especially those with many-to-many relationship) to:
  
  • Minimize redundant data elements.
  
  • Minimize awkward many-to-many relationships.
  
  • Increase stability and flexibility.

• **Referential integrity rules**

  • Used by relational databases to ensure that relationships between coupled tables remain consistent.

  • Idea: not add a record to the table with the foreign key unless there is a corresponding record in the linked table

  • E.g, Not add supplier # 8266 to PART unless there is a 8266 in SUPPLIERS
Entity-Relationship Diagram for the Database with Four Tables

This diagram shows the relationship between the entities SUPPLIER, ART, LINE_ITEM, and ORDER.

Figure 5-6
### The Final Database Design with Sample Records

#### Figure 5-5

<table>
<thead>
<tr>
<th>PART</th>
<th>LINE_ITEM</th>
<th>ORDER</th>
<th>SUPPLIER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Part_Number</td>
<td>Part_Name</td>
<td>Unit_Price</td>
<td>Supplier_Number</td>
</tr>
<tr>
<td>137</td>
<td>Door latch</td>
<td>22.00</td>
<td>8259</td>
</tr>
<tr>
<td>145</td>
<td>Side mirror</td>
<td>12.00</td>
<td>8444</td>
</tr>
<tr>
<td>150</td>
<td>Door molding</td>
<td>6.00</td>
<td>8263</td>
</tr>
<tr>
<td>152</td>
<td>Door lock</td>
<td>31.00</td>
<td>8259</td>
</tr>
<tr>
<td>155</td>
<td>Compressor</td>
<td>54.00</td>
<td>8261</td>
</tr>
<tr>
<td>178</td>
<td>Door handle</td>
<td>10.00</td>
<td>8259</td>
</tr>
<tr>
<td>Order_Number</td>
<td>Part_Number</td>
<td>Part_Quantity</td>
<td></td>
</tr>
<tr>
<td>3502</td>
<td>137</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>3502</td>
<td>152</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>3502</td>
<td>178</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Supplier_Number</td>
<td>Supplier_Name</td>
<td>Supplier_Street</td>
<td>Supplier_City</td>
</tr>
<tr>
<td>8259</td>
<td>CBM Inc.</td>
<td>74 5th Avenue</td>
<td>Dayton</td>
</tr>
<tr>
<td>8261</td>
<td>B. R. Molds</td>
<td>1277 Gandolly Street</td>
<td>Cleveland</td>
</tr>
<tr>
<td>8263</td>
<td>Jackson Components</td>
<td>8233 Micklin Street</td>
<td>Lexington</td>
</tr>
<tr>
<td>8444</td>
<td>Bryant Corporation</td>
<td>4315 Mill Drive</td>
<td>Rochester</td>
</tr>
</tbody>
</table>
Sample Order Report

Order Number: 3502  
Order Date: 1/15/2008

Supplier Number: 8259  
Supplier Name: CBM Inc.  
Supplier Address: 74 5th Avenue, Dayton, OH 45220

<table>
<thead>
<tr>
<th>Order_Number</th>
<th>Part_Number</th>
<th>Part_Quantity</th>
<th>Part_Name</th>
<th>Unit_Price</th>
<th>Extended_Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>3502</td>
<td>137</td>
<td>10</td>
<td>Door latch</td>
<td>22.00</td>
<td>$220.00</td>
</tr>
<tr>
<td>3502</td>
<td>152</td>
<td>20</td>
<td>Door lock</td>
<td>31.00</td>
<td>$620.00</td>
</tr>
<tr>
<td>3502</td>
<td>178</td>
<td>5</td>
<td>Door handle</td>
<td>10.00</td>
<td>$50.00</td>
</tr>
</tbody>
</table>

Order Total: $890.00
DBMS

- Specific type of software for creating, storing, organizing, and accessing data from a database
- Separates the logical and physical views of the data
  - **Logical view**: how end users view data
  - **Physical view**: how data are actually structured and organized
- **Examples of DBMS**: Microsoft Access, DB2, Oracle Database, Microsoft SQL Server, MySQL
Human Resources Database with Multiple Views

Figure 5-7
Operations of a Relational DBMS

• **Select:**
  - Creates a subset of all records meeting stated criteria

• **Join:**
  - Combines relational tables to present the server with more information than is available from individual tables

• **Project:**
  - Creates a subset consisting of columns in a table
  - Permits user to create new tables containing only desired information
The select, project, and join operations enable data from two different tables to be combined and only selected attributes to be displayed.
Capabilities of Database Management Systems

• Data definition capabilities:
  • Specify structure of content of database.

• Data dictionary:
  • Automated or manual file storing definitions of data elements and their characteristics, e.g., names, descriptions, size, type, format, etc.

• Querying and reporting:
  • Data manipulation language (add, change, delete and retrieve data)
    • E.g., Structured query language (SQL)
    • E.g., Microsoft Access query-building tools
Example of an SQL Query

SELECT PART.Part_Number, PART.Part_Name, SUPPLIER.Supplier_Number, SUPPLIER.Supplier_Name
FROM PART, SUPPLIER
WHERE PART.Supplier_Number = SUPPLIER.Supplier_Number AND Part_Number = 137 OR Part_Number = 150;

Illustrated here are the SQL statements for a query to select suppliers for parts 137 or 150. They produce a list with the same results as Figure 5-8.

Figure 5-10
Online Analytical Processing (OLAP)

- Supports multidimensional data analysis
  - Enable users to view same data in different ways using multiple dimensions
  - Dimension can be — product, pricing, cost, region, or time period
  - E.g., comparing sales in East in June versus May and July
Figure 5-14 Multidimensional Data Model

PROJECTED

ACTUAL

PRODUCT

Nuts

Bolts

Washers

Screws

REGION

East

West

Central
Data Mining

• Finds hidden patterns and relationships in large databases
• Types of information obtainable from data mining
  • Associations: occurrences linked to single event, e.g., chip & coke
  • Sequences: events linked over time, e.g., purchase new appliance with the first two weeks of new house
  • Classifications: patterns describing a group an item belongs to, e.g., discover characteristics of customers who are likely to leave the services.
  • Clusters: discovering as yet unclassified or not defined groupings
  • Forecasting: uses series of values to forecast future values through the pattern of data.
Data Mining

- **One popular use of data mining:** identifying profitable customers

- **Predictive analysis:**
  - Uses historical data, and assumptions about future conditions to predict outcomes of events
  - E.g. such the probability a customer will respond to an offer or purchase a specific product
• **Text Mining**
  - Unstructured data (mostly text files) accounts for 80 percent of an organization’s useful information.
  - Text mining -- extract key elements from, discover patterns in, and summarize large unstructured data sets.

• **Web Mining**
  - Discovery and analysis of useful patterns and information from the Web
Network Architecture

- Network architectures are layered
- Each layer
  - uses the services of the layers below
  - To offer more advanced services to layer above
- Allows layers to be designed independently
- We will talk about 3 layers next...

<table>
<thead>
<tr>
<th>Layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network</td>
</tr>
<tr>
<td>Link</td>
</tr>
<tr>
<td>Physical</td>
</tr>
</tbody>
</table>
What are some examples of communications networks?

- Public Telephone Network
- Internet
- LANs (Local Area Networks)
What does a network do?

1) Transport data from one host to another.
Physical Layer: Convey bits over a wire

Bits: 010110...

![Diagram showing voltage over time with and without noise](image-url)
Other schemes for mapping a bit sequence to a physical sequence are possible. These are called *modulation* schemes.
Link Layer

- **Make a Frame** link out of a bit link
  - Instead of endless sequence of 1s and 0s, we want distinct “packages” of data that are separate from each other
- **Say we want to send 2 Frames with data**
  - 01010101010111010 and 101010101011010
  - Concatenate them and send them as a sequence?
- **How can the receiver tell where the new frame begins?**
- **Solution:** insert a special sequence at the start of frame: for example: 01111110
Also does error detection/correction

- Insert extra information that helps the receiver to determine if the data has been corrupted.

- Example: parity bit
  - Sender adds a 1 or zero to the end of data so the number of ones is always odd.
    - 10011 or 10000
  - If receiver counts an even number of ones, than it knows the data was corrupted.
Want to allow multiple hosts to *share a link*

How do they avoid talking at the same time?

- Don’t transmit if you hear another host transmitting
- If there is a collision, stop wait a random amount of time, and try again
- This is a **Medium Access Control (MAC)** Protocol
How do the hosts on this Ethernet identify each other?

Each host (actually each interface)
- has a globally unique MAC address
- Cannot be changed
Ethernet Hub

- Hub broadcasts packets on a link to all others
- As if all hosts connected to single link
  - We say it is a Single collision domain
- Only one host can talk at a time
Ethernet Switch

- **If switch knows where the destination is**
  - Switch forwards an incoming frame to destination only.
  - Otherwise, it broadcasts it to everyone.
- **Thus, parallel conversations possible.**
Network Layer

- **A wants to send some data to C**
  - Suppose A knows C’s address
- **A sends a packet towards C**
  - A marks his packet with C’s Address (an **IP Address**)
Post Office Analogy

Bob in New York

Alice Smith
1156 High St
Santa Cruz 95064

NY Post Office

Plane to London

Plane to SFO

SFO Post Office

Plane to SFO

Truck to Santa Cruz

Santa Cruz Post Office

Truck to 41st Ave

Truck to High St

Truck to Santa Rosa

Alice

Look at beginning of zip code.
Make forwarding decision

Look at address
Make forwarding decision
Network Layer

Host A
128.114.60.200
(IP Address)

Host B
128.114.60.201

Host C
128.114.60.202

Host D
128.114.60.203

Link 1

Link 2

Link 3

Destination Address: 128.114.60.202 & size

Header

Payload Data
Network Layer

Host A 128.114.60.200

Host B 128.114.60.201

Host C 128.114.60.202

Host D 128.114.60.203

- A uses Link 1 to send to B
- B looks at
  - Packet Header
  - Routing Table

<table>
<thead>
<tr>
<th>Address</th>
<th>Next Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>128.114.60.202</td>
<td>Link 2</td>
</tr>
<tr>
<td>128.114.60.203</td>
<td>Link 3</td>
</tr>
</tbody>
</table>
Routing in the Internet

Many feasible paths from source to destination.
Routing

- Updating (dynamic) the routing table
- Objective: each packet gets closer to destination

Packet forwarding

- Transmitting each packet on the appropriate output link
- Based on routing table
Routing Algorithms

Routers talk to each other to build their routing tables

“I am accepting Traffic to 114.211.1.X”
## Routing Table

<table>
<thead>
<tr>
<th>IP Address</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>114.211.1.X</td>
<td>1</td>
</tr>
<tr>
<td>200.261.19.X</td>
<td>2</td>
</tr>
</tbody>
</table>

Diagram:
- Host A
- Host B
- Host C
- HUB

Connections:
- 114.211.1.1
- 114.211.1.2
- 114.211.1.3
- 114.211.1.4
Internet Routing is Hierarchical

- Backbone or NSP: (MCI, ATT)
- Autonomous System (AS)
- ISP or IAP (CRUZIO, AOL)
- Customer AS
- ISP
- AS
Routing Concerns

- Long routes
- Circular routes
- Hijacking routes
- Route flapping
IP Addresses vs Mac Addresses

- Hierarchical
  - The beginning bits tell you which network the host is on
  - Ex: UCSC addresses start with 128.114.X.X
  - The last bits tell you which host of the network

- Changeable
  - Changes with location of host

  - 4 bytes

- Not Hierarchical
  - Beginning bits tell nothing useful
  - Helpful in identify the equipments

- Not Changeable
  - 6 bytes
Link and Network Layer Interaction

Strip MAC header off frame. Forward IP packet based on Routing table.
Issues In Networking

- Sharing of Limited Resources
  - How Should A and B share a link with limited bit rate?
**Issues In Networking**

- **Time Division Multiplexing**
  - gives each connection the use of the link a fixed fraction of time
  - Fixed fraction of resources reserved for each connection
  - Technology called *circuit switching*.

- **Problem**
  - When A is silent, A’s fraction of link goes unused.
Issues In Networking

- **Statistical Multiplexing**
  - Link shared in such a way that connections are not assigned fixed fraction of Link.
  - $A$ and $B$ unlikely to offer peak rate at the same time.
  - $\max(A + B) < \max(A) + \max(B)$

![Diagram of network connections showing sources A and B, and destinations A and B. Graphs illustrating data flow and rate.]
Statistical Multiplexing

- Because resources aren’t reserved. It’s possible offered load too high.
- Packets are put into a queue.
- If offered load remains too high, queue will fill up and overflow.
Transport Protocols

- The Internet is unreliable
  - It will make a “best effort” to get your packet to its destination

- Packets can be lost because of
  - Congestion
  - Link errors
  - Routing problems

Diagram:

```
Application
Presentation
Session
Transport
Network
Link
Physical
```
Transmission Control Protocol (TCP)

- **Retransmit mechanism for reliability**
  - Receiver sends acknowledgements to sender
  - If a packet is lost, source fails to get ACK, and then retransmits.

- **Congestion control**
  - If congestion perceived (by lost packets)
  - Source reduces its send rate
    - When loss, sender reduces send rate by half
    - Otherwise slowly increases
TCP cont’d

- TCP port numbers
  - TCP Header has a “port” number field
  - Helps host sort out how to route packets to applications

<table>
<thead>
<tr>
<th>Your Computer</th>
<th>IP Header</th>
<th>TCP Header</th>
<th>Payload</th>
</tr>
</thead>
</table>

Port 80

Packet

Email Client

Port 80

Port 143
For some applications packet retransmissions are not worthwhile, e.g., real-time Skype

Why?

For those applications, we use UDP

UDP is a transport protocol that

- Does not do retransmissions
- Does not do congestion control
When networks are congested, certain sessions (source-destination pairs) should reduce offered rates.
- Today all TCP sessions slow down when they detect packet losses.
- UDP sessions do not slow down.

What are some alternative strategies?
- Have those whose applications aren’t as sensitive slow down more?
  - How would we know which are less sensitive
Pricing within the Internet

- **Customer pays an ISP**
  - Often Flat Rate per month
- **ISP pays a backbone AS**
  - Often just flat rate, dependent on access link speed.
  - Sometimes based on total usage
- **Backbone NSPs peer with each other**
  - Often for free if they exchange comparable amounts of traffic.
- **Overall...**
  - Internet billing today is much more course grained than telephone billing.
The Global Internet

The World Wide Web

- **HTML (Hypertext Markup Language):**
  - Formats documents for display on Web

- **Hypertext Transfer Protocol (HTTP):**
  - Communications standard used for transferring Web pages

- **Uniform resource locators (URLs):**
  - Addresses of Web pages
  - E.g., http://www.megacorp.com/content/features/082602.html
Domain Names

Purpose: maps name to addresses
IP addresses are inconvenient for people
- 32 bits hard to remember
- 128 bits very hard to remember
- Subject to scalability issues

Domain names
- e.g. ucsc.edu
- Easier to remember than IP addresses
- However, we need some way of mapping domain names to IP addresses.
Domain Name System (DNS)

- Root Name Server
- Berkeley (berkeley.edu) Name Server
- UCSC (uscs.edu) Name Server
- EECS (eecs.berkeley.edu) Name Server
- SoE (soe.ucs.edu) Name Server
Hierarchy in Addresses vs. Names

Addresses hierarchical in topology

- Maximize “wild cards” and distribute address administration

Names hierarchical in administration

- Single administered organizations often distributed topologically (e.g. ibm.com)
Transport Protocols

- The Internet is unreliable
  - It will make a “best effort” to get your packet to its destination

- Packets can be lost because of
  - Congestion
  - Link errors
  - Routing problems
OSI (Open Systems Interconnects) Layers

<table>
<thead>
<tr>
<th>Layer</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Internet Explorer, Outlook Email, Real Player, …</td>
</tr>
<tr>
<td>Presentation</td>
<td>TCP, UDP</td>
</tr>
<tr>
<td>Session</td>
<td>Internet Protocol (IP), …</td>
</tr>
<tr>
<td>Transport</td>
<td>Ethernet, Wi-Fi, SONNET, …</td>
</tr>
<tr>
<td>Network</td>
<td>Modulation Schemes: QAM, OFDM, etc…</td>
</tr>
<tr>
<td>Link</td>
<td></td>
</tr>
<tr>
<td>Physical</td>
<td></td>
</tr>
</tbody>
</table>
Some Typical Topologies

Home Network

Diagram:
- Two computers
- Ethernet Switch
- Router
- DSL Modem
- Telephone Line
- (to local Office)
Small/Medium Business

- Router with Firewall
- T1 Modem
- Ethernet Switch
- Web Site Server
- T1 Line

To Local Office
Network Service Provider

Network Access Point

Network Access Point

Network Access Point
Large E-Business

Incoming HTTP Requests

Interconnected with Gigabit Ethernet or other technology

Logic Flow of Interaction

Logical
Presentation Logic
(Assembling Web page)

Load Balancer

Application Servers

Web Servers

Databases

Customers

Merchandise

Orders
Web Caching

- Speed up web page loading by storing previously seen components locally

http://www.ucsc.edu

Cache on Hard Drive

Web server
The Global Internet

The World Wide Web

- **Search engines**
  - Started in early 1990s as relatively simple software programs using keyword indexes
  - Finding useful information on the Web nearly instantly.

- **Search engine marketing** – major source of revenue
  - Keyword auctions
How Google Works

1. User enters query

2. Google’s Web servers receive the request. Google uses an estimated 450,000 PCs linked together and connected to the Internet to handle incoming requests and produce the results.

3. Request is sent to Google’s index servers that describe which pages contain the keywords matching the query and where those pages are stored on the document servers.

4. Using the PageRank software, the system measures the “importance” or popularity of each page by solving an equation with more than 500 million variables and two billion terms. These are likely the “best” pages for the query.

5. Small text summaries are prepared for each Web page.

6. Results delivered to user, 10 to a page.

Figure 6-13: The Global Internet
Google is the most popular search engine on the Web, handling 56 percent of all Web searches.
The Global Internet

The World Wide Web

**Web 2.0**

- Refers to more *interactive Internet-based services* enabling people to collaborate, share information, etc.

- **Blogs**: chronological, informal Web sites created by individuals using easy-to-use Weblog publishing tools *(readers cannot change it)*

- **RSS (Really Simple Syndication or Rich Site Summary)**: syndicates Web content so content can be automatically placed into another setting, *keep up with your favorite sites without constantly checking them*

- **Wikis**: collaborative Web sites where visitors can add, delete, or modify content on the site *(readers can also be writers)*
Cellular systems

- 2G -- Competing standards for cellular service
  - United States: **CDMA (Code Division Multiple Access)**, e.g., Verizon, Sprint
  - Most of rest of world: **GSM (Global System for Mobile Communication)**, e.g., T-Mobile, ATT (compatible for roaming)

- Third-generation (3G) networks (fast speed)
  - UMTS (GSM extension) ATT
  - CDMA 2000

- 4G
  - LTE (Long Term Evolution), WiMax
Final Quiz:

Please list two principles of “layering” in designing architecture of network computing.

Good Luck with the final.