Application Architecture
- **Decomposition**: Divide the architecture into interacting modules.
- **Assembly**: Find subsystems available for purchase.
- Most architecture design is a mixture of decomposition and assembly.

Decomposition Example
- Example: manage bank accounts
  - Decompose into software modules for
    - transaction processing,
    - statement generation
  - Further decompose transaction processing module into deposit and withdraw modules...

Assembly Example
- example - ecommerce platform
  - Acquire
    - Linux pc (application server)
    - IBM Mainframe (data server)
    - Oracle DBMS
    - Apache Web Server Software
  - Assemble all pieces together.
  - Mix with custom developed application logic module.

Object-Oriented Architectures
- Object-Oriented Programming (OOP) Languages
  - C++
  - Java
  - Smalltalk
- The basic unit of modularity in OOP is an object.

Objects
- Example: Bank account
  - has a balance of $5000
  - belongs to Joe Schmoe
  - is a checking account
  - can have money deposited to it
  - can have money withdrawn from it

Attributes

Behaviors
Objects

- An **attribute** is a numerical value or data that is externally visible, and may be changeable.
  - Ex: The bank account's balance is $5000
- A **method** is an action available at the object interface
  - Other objects invoke method, pass parameters and get returned data or other objects.
  - We can invoke the "check_balance" method and get returned the number $5000

Object Classes and Instances

- Some objects share types of attributes and methods.
  - They have the same **class**
- **Example**
  - Class: Bank_Account
  - Instances:
    - Schmoe_Account  balance: $5000
    - Smith_Account  balance: $10000
  - Each instance is a separate object with its own data

Declaring Classes

**When we program, we define or “declare” each class we plan to use.**
- **Example**: We plan to use a class called "bank_account"
  - It will have the attributes: balance, owner, etc...
  - It will have the methods: check_bal, withdraw, deposit, ...
  - Later on we fill in the details of what each method does.
- Once we declare a class, we can create instances of it.
  - Schmoe_account, smith_account, etc.

Method Invocation

- **Objects communicate with each other by invoking each other's methods**
  - **(method invocation)**
  - ATM Object
  - Schmoe_Account
    - Attributes:
      - balance: $5000
    - Methods:
      - withdraw ( )
      - check_bal ( )
  - $4500

  - **Terminology**:
    - Client object -- object invoking the method
    - Server object -- object whose methods are being invoked

Software Objects

- **In OOP an object can**
  - Represent a real world entity
    - Bank account
  - Be a proxy of a real world entity
    - Owner of a customer
    - Other software talks to proxy using method invocations
  - Model a real-world entity
    - For purposes of simulation
    - Motion of a train

ORDBMS

- **Earlier in the class we talked about relational DBMS**
  - The most common database management system that organizes data into tables.
- **ORDBMS (Object Relational DBMS)**
  - Retrieve and store object instance data in a relational database
Remote Method Invocation

- Sometimes we want to allow an object to invoke methods on an object located on another machine.
- This is called Remote Method Invocation (RMI)
- Doing this requires middleware called
  - Distributed Object Management (DOM)

Software Reuse

- Size and complexity of applications growing dramatically
- In order to contain costs, we need to be able to reuse pieces of software
- Reuse is difficult. Why?
  - OOP was developed in part to promote re-use, but has had limited success in that regard.

Software components

- Software components are reusable modules that can be bought from outside vendors.
- How is a component different from an object?
  - More importance on
    - Encapsulation
    - Well defined and documented interfaces

Component Assembly Tools

- Visual or integrated development environment (IDE)
  - MS Visual Studio
  - IBM Visual Age
  - Symantec Visual Café
- Scripting Assembly - Text based
  - TCL
  - Perl
  - JavaScript

Software Frameworks

- A preexisting architecture and library of components from a common vendor to help developers
- Enables reuse, and ensures component interoperability.
- Examples:
  - Sun J2EE/Java Beans
  - Microsoft .Net
  - Adobe Flash, Microsoft Silverlight

AA Case
American Airlines Case – Systems Operations and Control center (SOC)

- **Flight Dispatching*** - focus of case
  - Flight Path
  - Fuel Load
  - en route weather, problems
  - Each dispatcher assigned a geographic area
- **Load Planning**
  - optimize loading of passengers and freight
  - consider runway length, weather, plane type, etc.
- **Crew Scheduling**
  - Crews under strict regulation about amount of time can work
  - Certain crews can fly certain planes
  - seniority
  - positioning for future flights

### Dispatch Automation Package

- **Flight tracking application**
  - **View 1**
    - List of all flights dispatcher responsible for
  - **View 2**
    - Dependencies of one flight on other flights.
- **Message tracking**
  - e-mail to flight crew

### View 1:

<table>
<thead>
<tr>
<th>Flight Tracking App</th>
<th>Flight Tracking App</th>
<th>Flight Tracking App</th>
</tr>
</thead>
<tbody>
<tr>
<td>View 1:</td>
<td>View 2:</td>
<td>View 3:</td>
</tr>
<tr>
<td>List of all flights dispatcher responsible for</td>
<td>Dependencies of one flight on other flights.</td>
<td>e-mail to flight crew</td>
</tr>
</tbody>
</table>

### AA IT Architecture

- **Flight Tracking Application**
  - Fall 90 - Built as prototype as a way for someone to teach himself OOP
  - May 91 - OK to develop application
  - Work divided (one person in charge of each)
    - User interface
    - data model
    - data exchange with FOS
  - Nov 91 - Production installation complete
  - 2 months testing
Flight Tracking Application

**Facts and Figures:**
- Written in Smalltalk
- 210 classes
- 2000 methods
- 160000 lines of assembler code
- 150000 object instances in memory at all times

**OOP + good architecture made 3 changes easier**
- Changed how flight was referenced, major change to data model
  - (1.5 weeks)
- Introduced File servers to cache FOS data
  - (1 day, 4 weeks test)
- Developed message queuing monitor
  - (1 wk, test 3 wk)

**Good architecture allowed extensions later**
- Feature to allow dispatcher to focus on very really late flights
- Flight lock - stop flights to airport for bad weather
- In flight fuel calculation
- Geographical flight monitor

Did AA follow Application Lifecycle Model?

- Fall 90 - Built as prototype as a way for someone to teach himself OOP
- May 91 - OK to develop application
- Work divided (one person in charge of each)
  - User interface
  - Data model
  - Data exchange with FOS
- Nov 91 - Production installation complete
  - 2 months testing
  - Extensions added later

Information access

by

David G. Messerschmitt
A hierarchy

**Data:** numbers, character strings, etc.

**Information:** recognizable patterns organized so as to inform or influence us in some way

**Knowledge:** concepts, relationships, truths, principles.

**Wisdom:** insight or judgement

---

Classify these

- "XV", "SF", 34, "CN", 16
- The 49-ers won Super Bowl XV by a score of 34 to 16.
- The National Football Conference wins 17 out of 20 Super Bowls on average.
- The best team usually wins.

---

Classify these

**Relative to A Streetcar Named Desire:**
- Tennessee Williams
- Actor
- Critic
- Playbill magazine

**Relative to Understanding Networked Applications:**
- D.G. Messerschmitt
- Morgan Kaufmann
- Amazon.com

---

Roles in information access

- **Author or publisher**
- **User**
- **Indexer or organizer**
- **Librarian or teacher or interpreter**
- **Recommender**

---

Exercise

User
Author or publisher
Indexer or organizer
Librarian or teacher
Recommender

How are these roles being changed by networked computing?

---

Push vs. pull

User
Control over what is provided
Time when it is provided

Intermediate cases:
- Notification
- Subscription

Publisher
Push
Pull
Proper roles of push and pull in a workgroup

<table>
<thead>
<tr>
<th>Pull: work</th>
<th>Push: attention</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brainstorming</td>
<td>Notification of topic</td>
</tr>
<tr>
<td>Accessing documents</td>
<td>Notification of document availability</td>
</tr>
<tr>
<td></td>
<td>Reminder of deadlines</td>
</tr>
</tbody>
</table>

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

Question

What are some differences between push and pull with respect to:
- invasiveness on the user?
- refinement of the information received?
- timeliness with which information received?

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

Some modalities of information access

Aids in finding useful information

Besides the information content itself, other aids:
- reference to related information: hyperlink
- list of content: index
- description of content: metadata
- judgment of content: recommendation

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

Exercise

Give an example of the following functions in the context of movie rentals:
- Hyperlink
- Index
- Metadata
- Recommendation

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

Question

Comment on the following widely held beliefs (at their time):
- "the movie will displace legitimate theater"
- "television will displace movies"
- "remote learning will displace the university campus as we know it"

What does this suggest about networked applications?

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