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- can have money withdrawn from it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

invoke: check_bal()

return: $5000

Invoke: withdraw ( $500)

return: “successful”

Terminology:
- Client object -- object invoking the method is the
- Server object - object whose methods are being invoked
In OOP an object can
- Represent a real world entity
  - Bank account
- Be a proxy of a real world entity
  - Proxy of a customer
  - Other software talks to proxy using method invocations
- Model a real-world entity
  - For purposes of simulation
  - Motion of a train
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 regulations 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
## Flight Tracking FD01

<table>
<thead>
<tr>
<th>CRIT</th>
<th>C/P</th>
<th>F/A</th>
<th>A/C</th>
<th>O</th>
<th>TYPE</th>
<th>FROM</th>
<th>FLT/DA</th>
<th>DPT</th>
<th>SKED</th>
<th>LAT</th>
<th>L AVL</th>
<th>LAT</th>
<th>S DLAY</th>
<th>QPAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Htl</td>
<td>Htl</td>
<td>495</td>
<td>SP80</td>
<td>1655</td>
<td>1459/11</td>
<td>OMA</td>
<td>1355</td>
<td>1406</td>
<td>ORD</td>
<td>1517</td>
<td>0 2 53</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Orig</td>
<td>Orig</td>
<td>517</td>
<td>SP80</td>
<td>Thru</td>
<td>632/11</td>
<td>STL</td>
<td>1415</td>
<td>1413</td>
<td>DFW</td>
<td>1607 C</td>
<td>5 2 54</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Orig</td>
<td>Orig</td>
<td>495</td>
<td>SP80</td>
<td>Thru</td>
<td>1226/11</td>
<td>DFW</td>
<td>1412</td>
<td>1417</td>
<td>BOS</td>
<td>1700</td>
<td>10 2105</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>345</td>
<td>345</td>
<td>110</td>
<td>7271</td>
<td>783</td>
<td>1640/11</td>
<td>DFW</td>
<td>1418</td>
<td>1446</td>
<td>ORD</td>
<td>1755 C -15 2 54</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>991</td>
<td>991</td>
<td>495</td>
<td>SP80</td>
<td>361</td>
<td>189/11</td>
<td>LAX</td>
<td>1422</td>
<td>1426</td>
<td>ORD</td>
<td>1817</td>
<td>0 3 79</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>600</td>
<td>1012</td>
<td>517</td>
<td>SP80</td>
<td>901</td>
<td>632/11</td>
<td>STL</td>
<td>1435</td>
<td>1447</td>
<td>DFW</td>
<td>2007 C</td>
<td>5 3 32</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>444</td>
<td>444</td>
<td>279</td>
<td>SP80</td>
<td>1832</td>
<td>1216/11</td>
<td>DFW</td>
<td>1435</td>
<td>1484</td>
<td>Show Legend</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Htl</td>
<td>Htl</td>
<td>464</td>
<td>7271</td>
<td>1655</td>
<td>1610/11</td>
<td>DFW</td>
<td>1439</td>
<td>1484</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Status:** Last update time 16:03:04 GMT

*Note: This figure is a reconstruction, all data is fictional.*
View 2:
AA IT Architecture

"Passenger Service System" (Does Reservations)

"Fare Pricing Complex" Responsible for updating Fares (3 mainframes)

SABRE LAN

"Flight Operating System" Maintains Critical Data 1-Mainframe

AMR Intranet

SABRE SOC

File Servers (Cache FOS data every 2 min)

Dispatcher Workstations

Dispatcher automation package
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
Flight Tracking Application

- 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)
Flight Tracking Application

- 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

1) CONCEPTUALIZATION
2) ANALYSIS?
3) ARCHITECTURE
4) DEVELOPMENT
5) TESTING
6) DEPLOYMENT
7) OPERATION, EXTENSION, MAINTENANCE
Information access

by

David G. Messerschmitt
Copyright notice

©Copyright David G. Messerschmitt, 2000. This material may be used, copied, and distributed freely for educational purposes as long as this copyright notice remains attached. It cannot be used for any commercial purpose without the written permission of the author.

Adapted from slides for Understanding Networked Applications
By David G Messerschmitt. Copyright 2000. See copyright notice
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 Bowl’s on average.
- The best team usually wins.

Adapted from slides for *Understanding Networked Applications*  
By David G Messerschmitt. Copyright 2000. See copyright notice
Roles in information access

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

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

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

User

Author or publisher

Indexer or organizer

Librarian or teacher

Recommender

How are these roles being changed by networked computing?

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

User

Control over what is provided
Time when it is provided

Intermediate cases:
Notification
Subscription

Publisher

Push

Pull

Adapted from slides for Understanding Networked Applications
By David G Messerschmitt. Copyright 2000. See copyright notice
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?
Some modalities of information access

- **Pull**
  - Search, navigate, browse
  - Delegate
  - Intermediary
  - Agent

- **Push**
  - Subscribe
  - Aggregate, filter, consolidate

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

Give an example of the following functions in the context of movie rentals:

Hyperlink
Index
Metadata
Recommendation
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?