Announcements

- Database Lab in BE 109 (you have attend one)
  - Thursday, 2/17/11 ONLY, 12:30-2:00PM
  - Friday, 2/18/11 ONLY, 9:00-10:00am
  - Saturday, 2/19/11 ONLY, 12:00-2:00pm
- Assignment 3 out
  - DUE next Thursday, 2/17 submit to eCommons
- Next time read:
  - Messerschmitt Ch. 7

Application Architecture Design

- The most important step against complexity
  - Hardest to change
  - Influences everything that follows
- Conceptualization
  - What is it you are trying to do?
- Example Concept:
  - Small HHC for flight attendants
  - HHC tells flight attendants which passengers are higher priority
  - Who paid the highest fares
  - Who has been a more valuable customer in past
  - Flight attendant discriminates based on this
    - Free drinks, meals, and pillows to valuable customers

Example Concept:

```
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

Architecture

- What is the complexity of such a problem?
- How do you begin to architect a solution for a problem like this?
- Follow the principle that says: Break it into modules!
- What is a "good" architecture?
Architecture

HHC Architecture

Each Tier is decomposed into modules.

When a module is composed of sub-modules, the architecture is **hierarchical**.

HHC Application

Palm OS

Networking Infrastructure

User Interface

Coordination

With HHC Server

Data

Management

We are using a **layered architecture** as well.

Allows reuse of previously built infrastructure.

Some aspects of software complexity

1) # of elements increases $\rightarrow$ system's complexity increases
2) problem domain is complex
3) a lot of constraints
4) every case must be foreseen
5) continuous vs discrete, not exhaustively check every case
6) team effort
7) integration of different parts

Properties of Modularity

- (idea: divide into smaller parts and deal with each part separately)
  - Functionality
  - Hierarchy
  - Separation of concerns
  - Interoperability
  - Reusability

Granularity tradeoff

- How big should we make the modules
  - Many simple small ones… (fine granularity)
  - Or a few complicated big ones… (coarse granularity)
- Which is better?
  - Coarse granularity: Fewer things to keep track of
  - Fine granularity: Modules are simpler and easier to understand
- **Hierarchy** of modules: Allows the system to be viewed at different granularity levels.
**Student Talks**

- Max Silveus Bus Proj: Coca-Cola
- Joy Siao Bus Proj: Starbucks

**Interfaces**

<table>
<thead>
<tr>
<th>PARAMETERS</th>
<th>Compute Mean and Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>N numbers of float type</td>
<td>2 Numbers of float type that signify: Mean, Variance</td>
</tr>
</tbody>
</table>

The data passed through an interface have 3 properties:
1. Name (e.g., employee_name)
2. Type (e.g., string)
3. Value (e.g., "John Smith")

**Implementation**

**Module B**

- Compute Mean and Variance
- Computation of key statistics

**Implementation 1:**

\[
\text{SUM} = \sum_{i=1}^{N} x_i \\
\text{MEAN} = \frac{\text{SUM}}{N} \\
\text{VARIANCE} = \sum_{i=1}^{N} (x_i - \text{MEAN})^2
\]

**Module A**

- Computation of key statistics

**Implementation 2:**

Though different, this implementation is ok too.

We can choose the implementation details however we want, as long as we comply with the agreed interface.

**Encapsulation**

**Module B**

- Compute Mean and Variance
- Computation of key statistics

**Implementation 2:**

- Should he use it?
  - NO!!! Why??
  - Either A should compute "SUM" itself, or the interface of B should be redesigned

**Module A**

- Computation of key statistics

"I need to get the sum. I'll just take it from B"
Encapsulation

- The designer of B might take measures to hide "SUM" from A so that A is not able to violate the agreed interface.
  - Example: B does not declare "SUM" as a global variable.

- Making a modules implementation details inaccessible to other modules is called **encapsulation**

Interfaces

- Module A
  - Parameters:
    - N numbers of float type

- Module B
  - Parameters:
    - 2 Numbers of float type that signify:
      - Mean
      - Variance

- Computation of key statistics

- Interface

- Returns

- This simple interface example allows for only one action of module B.
  - Action is "Compute mean and variance."

- Other examples are possible.

Possible software interface

Menu of actions

- action-1
- action-2
- action-3
- ...

Example:

- Action 1: Compute mean
- Action 2: Compute variance
- Action 3: Compute sum
- Etc..

Protocol

- In addition to atomic actions, an interface may define protocols
  - Protocol == finite sequence of actions required to achieve a higher level function
  - One action can be shared by multiple protocols
  - Multiple modules may participate in a protocol

Protocol Example

- HHC Server
  - Hello: I'm the HHC of Airplane#1234
  - Hello: I'm the gate 32 server
  - Tell me about the passengers of my next flight
  - Return Passenger data
  - Tell me about the weather at my next destination
  - Return Weather data

- HHC

Another Interface Example:

- **Automatic teller machine (ATM)**

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

1. **Identify interface building blocks**

2. **Define available actions**

3. **Define, for each higher level function, a protocol**
   - Single action or a finite sequence of actions

### 1. 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

### 2. ATM actions

- **Authentication**
- **Account specification**
- **Amount specification**

#### Action: authentication

**Parameters**
- Internal functionality

**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

Returns
- None

**Action: amount**

Parameters
- Dollars_and_cents (typed on keypad)

Internally, amount noted

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?

**Goals**

Understand better
- how layering is used in the infrastructure
- how it contains complexity
- how it coordinates suppliers
- how it allows new capabilities to be added incrementally

More on layering

by

David G. Messerschmitt
**Interaction of layers**

Layer above is a client of the layer below.

Each layer provides services to the layer above…

…by utilizing the services of the layer below and adding capability.

Layer below as a server to the layer above.

---

**Example 1**

Bob sends a letter to Alice.

- Bob
- Envelope
- US Postal Service
- Shipping Container
- UK Royal Mail
- ABC Airlines
- Alice
- Envelope
- Shipping Container

---

**Three types of software**

- Application
  - Components and frameworks: What is in common among applications
  - Infrastructure: Basic services (communication, storage, concurrency, presentation, etc.)

---

**Major layers**

- Applications
- Application frameworks and components
- Middleware
- Operating system
- Network
- Infrastructure

---

**Layering**

- Elaboration or specialization
- Existing layers
- Layering builds capability incrementally by adding to what exists.

---

**Data and information**

- Application: Deals with information
  - Assumes structure and interpretation

- Infrastructure: Deals with data
  - Ignores structure and interpretation
### Example 2

<table>
<thead>
<tr>
<th>Application</th>
<th>Operating system</th>
<th>Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web server</td>
<td>File system</td>
<td>File</td>
</tr>
<tr>
<td>Web browser</td>
<td>Message</td>
<td>Message</td>
</tr>
<tr>
<td>Screen</td>
<td>Collection of packets</td>
<td>Assembly</td>
</tr>
</tbody>
</table>

### Package = file or message

Infrastructure deals with a package of data (non-standard terminology)
- collection of bits
- specified number and ordering

Infrastructure stores and communicates packages while maintaining data integrity
- File for storage
- Message for communication

### Data integrity

- Nothing is lost/changed in the representation/recovery of information
- Retain the
  - values
  - order
  - number
  - of bits in a package
- Also applies to more complicated forms of representation and data processing
  - E.g. Data Integrity in Databases

### Example 3: Network Infrastructure Expanded

- HHC Server Application
- Passenger Information
- Windows OS
  - TCP transport layer
  - WiFi Link Layer
  - WiFi Physical Layer
  - Networking Infrastructure

- HHC Client Application
- Palm OS
  - TCP transport layer
  - WiFi Link Layer
  - WiFi Physical Layer
  - Networking Infrastructure

- Airline Dataserver
- “Send me today’s flight information”

- DBMS
  - Unix OS

- Collection of Packets
- Networking Infrastructure (Contains: TCP/IP, WiFi)
Data and information in layers

- The infrastructure should deal with data, or at most minimal structure and interpretation
- The application adds additional structure and interpretation
- This yields a separation of concerns

Information in the infrastructure

- Sometimes it is appropriate for the infrastructure to assume structure and interpretation for data
  - to add capabilities widely useful to applications
  - to help applications deal with heterogeneous platforms, where representations differ
- Data types