Announcements

- 2nd Folio Article due today
- Assignment 4 out, (due Thursday 11/20)

Read
- Messerschmitt Ch 11.1 - 11.2 (325-335)
- Messerschmitt Ch 15.1 - 15.2 (415-425)

Presenters for Thursday 11/13
- Katherine Beeskau (Business paper)
- Alba Beltran (Business paper)
Student Talks

Omar Calles (Business paper)
Blake Irby (Business paper)
Sun Case
(continued)
Sun N-tier case

- What does Sun make?
  - Workstations
  - Servers
  - Software
Microsoft mid to late 90s

- Dominated Desktop software
  - Users familiar with Windows, Office, etc.

- NT servers
  - Fine for small intranets, “not industrial strength”
Sun N-Tier Case

- What is Java?
  - Programming Language
  - Portable between computers with different operating systems
  - Easy to write programs in
  - Easier re-use
  - But, programs are slow
What problems did the micro era produce?

- Desktops are expensive to maintain
  - TCO for windows PC $9900!

- Every PC had a lot of software that had to be maintained
  - Office, Windows, etc...

- Small differences, like the order in which software is installed, could make different PCs behave differently!
Sun’s Vision

- Thin Client model.
- Application Servers with Applications written in Java.
- NCs could retrieve applications from application server as needed.
- Applications compatible with any NC hardware and OS.
- Applications could be fixed, added, updated at the server level, rather than maintaining each PC.
Microsoft Vision

- Keep “fat-client” model
- Add some features to Windows to reduce administration costs
**SUN 3 - Tier**

Exhibit 1  Three-tier Architecture

**Tier One**
- Client Applets
- Client Hardware

**Tier Two**
- Applets
- App Server
- These had to be managed locally. If code needed to be updated, each app server had to be shut down, updated and rebooted

**Tier Three**
- Database
- JDBC
- Applets
- App Server

---

JDBC: Stands for Java Database Connectivity. It is a programming interface that lets Java applications access a database via the SQL language. RMI: Stands for Remote Method Invocation. It is the method by which a remote Java object from one location can be invoked from other Java virtual machines. HTTP: Stands for HyperText Transport Protocol. It is the communications protocol used to connect to servers on the World Wide Web.
Sun N-Tier

**Exhibit 3**  How the N-tier Architecture Works
Sun's Performance

Net Revenue

Net income
Today

- 3-tier model common.
- Sun’s version of 4-tier model not-common.
- N-tier model where Webserver and Application Server on separate equipment also common.
- Sun’s hardware business not strong.
  - Linux on cheap PCs most common servers
  - Microsoft desktops replacing Sun workstations
Today

- **Java**
  - Common in Server implementations
    - Example: Java Servlet implementing application logic in a banking application.
  - Often used to push simple applets onto client
  - Not common
    - For “big” desktop applications
    - Office Suite in Java not popular
  - Microsoft is still in business...
What could have Sun done?

- Compete on price with cheap PC servers running Linux?
- Sell a fat-client workstation that runs Windows and is price competitive with Dell, HP PCs, etc...
- Sell workstations at a price premium over PCs, focus on software reliability, run some Microsoft application, build brand cachet.
- Focus on Java based software and IT services for enterprises, withdraw from low-end hardware...
- Something else?
Architecture Example
**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
  - Ignore less valuable customers
Example Concept:
Architecture

- How do you begin to architect a solution for a problem like this?
- Break it into modules!
Architecture

HHC Server

Wireless Link

HEADQUARTERS

Airline Intranet

Airline Dataserver

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

We are using a *layered architecture* as well.
- Allows reuse of previously built infrastructure.
Granularity tradeoff.

- How big should we make the modules
  - Many simple small ones
  - Or a few complicated big ones...

- This aspect of modularity is called **granularity**.

- Which is better?
Again, we see layering and hierarchy. Between each module we specify an **interface**.

Standard Database “queries” (SQL) relayed to DBMS via OS and infrastructure.
Our architecture makes use of the existing interface of the airline database, so we don’t need to redesign it!
A simple interface: from within our HHC Server Architecture

- Compute Mean and Variance
- List of numbers
- Computation of key statistics
- Mean, Variance

- HHC Application
- Palm OS
- Networking Infrastructure
- Communication with airline database
- Communication with HHC
- Computation of key statistics
Interface specifications are often made precise by using data types.

- Example type: float
  - A number with a decimal place
  - Has a certain allowable range, and precision.
More on Data types

- Data passing an interface is often specified in terms of a limited number of standard data types.

- Data type = range of values and allowable manipulation.

- Data type does not presume a specific representation, to allow heterogeneous platforms.
  - Representation must be known when data passes a specific module interface.
Example data types

**Integer**
- “natural number between -32,767 and +32,768”
- Could be represented (in many ways) by 16 bits
  - since \(2^n = 65,536\)

**Float**
- “number of the form \(m \times 10^n / 32768\), where \(m\) is in the range -32,767 to +32,768 and \(n\) is in the range -255 to +256”
- Could be represented by 16+8 = 24 bits
More data types

Character

- “values assuming a-z and A-Z plus space and punctuation marks”
  - could be represented by 7 or 8 bits

Character string

- “collection of $n$ characters, where $n$ is customizable”
  - could be represented by $7^n$ bits
Compound data types

Programmer-defined composition of basic data types

Example:

```java
Employee {
    String name;
    String address;
    Integer year_of_birth;
    etc.
}
```
Interfaces

PARAMETERS

N numbers of Float type

Compute Mean and Variance

Computation of key statistics

2 Numbers of float type that signify: Mean, Variance

INTERFACE

RETURNS
One module should not be concerned with other module’s implementation

→ “Separation of concerns.”

One module should see the other only through its interface - implementation details hidden.

→ Abstraction
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.
Implementation

Module A

Computation of key statistics

Module B

Compute Mean and Variance

\[ X_i, i=1..N \]

\[ \text{MEAN, VARIANCE} \]

Implementation 1:

\[ \text{SUM} = \sum_{i=1}^{N} x_i \]

\[ \text{MEAN} = \frac{\text{SUM}}{N} \]

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

- Should he use it?
  - NO!!!! Why??

- Either A should compute “SUM” himself, or sit down with B and redesign the interface
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**.