Sun Case (continued)

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
**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

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

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?

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
Mean, Variance

Computation of key statistics

HHC Application
Palm OS
Networking Infrastructure
Communication with HHC

Interfaces

Compute Mean and Variance

N numbers of Float type

2 Numbers of float type that signify:
Mean, Variance

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 $2^{16} = 65,536$

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 $7 \times n$ bits

Compound data types

Programmer-defined composition of basic data types
Example:

Employee {
  String name;
  String address;
  Integer year_of_birth;
  etc.
}

5
Interfaces

PARAMETERS

N numbers of Float type

Computation of key statistics

Compute Mean and Variance

IMPLEMENTATION

RETURNS

Computation of key statistics

2 Numbers of float type that signify: Mean, Variance

Implementation

Module B

Compute Mean and Variance

Module A

Computation of key statistics

Implementation 1:

Implementation 2:

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

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.

Interfaces

PARAMETERS

N numbers of Float type

Computation of key statistics

Compute Mean and Variance

IMPLEMENTATION

RETURNS

Computation of key statistics

2 Numbers of float type that signify: Mean, Variance

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

Example:
Action 1: Compute mean
Action 2: Compute variance
Action 3: Compute mode
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

Protocol Example: Automatic teller machine (ATM)

What is the interface between this machine and the customer?

Steps

Define available actions
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
- Identity (card in slot)
- Institution (card in slot)
- PIN (typed on keypad)

Internal functionality: 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
- Account (touch screen from menu of choices)

Internal functionality: 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!

---

**More on layering**

by David G. Messerschmitt

---

**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

---

**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

---

**Layering**

Elaboration or specialization

Existing layers

Layering builds capability incrementally by adding to what exists
**Three types of software**

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

**Part of Microsoft vs. DOJ dispute**

<table>
<thead>
<tr>
<th>Microsoft position</th>
<th>DOJ position</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Infrastructure</td>
</tr>
<tr>
<td>Components and frameworks</td>
<td>Infrastrucure</td>
</tr>
</tbody>
</table>

**Major layers**

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

**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

**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
Data integrity

- Retain the values
- Order
- Number of bits in a package

Example 1

Bob sends a letter to Alice

US Postal Service

UK Royal Mail

ABC Airlines

Example 2

Application

Web server

Web page

Screen

HTML

Example 3

HHC Server Application

HHC Client Application

Windows OS

Palm OS

Networking Infrastructure

TCP/IP

WiFi

Example 3: Network Infrastructure Expanded

HHC Server Application

HHC Client Application

Windows OS

Palm OS

Networking Infrastructure

TCP/IP

WiFi

Example 4

HHC Server Application

DBMS

Windows OS

Networking Infrastructure

TCP/IP

WiFi
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

At most, data types

Data and information

- Application
  Deals with information
  Assumes structure and interpretation
  Assumes standard data types

- Infrastructure
  Deals with data types