Announcements

- Read Messerschmitt Ch 7 for Tuesday.
- Presenters for Tuesday
  - Melinda Hsieh
  - Salvador Barrios

Student Talks

- Chi Hou Ip
- Andrew Hale

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's Performance

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
- 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...
**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

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

**HHC Architecture**

- **HHC Application**
  - Palm OS
  - Networking Infrastructure
- **Coordination** (With HHC Server)
- **User Interface**
- **Data Management**

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

**Architecture**

- **Airline Dataserver**
- **HEADQUARTERS**
- **Intranet**
- **Wireless Link**

**HHC Architecture**

- **HHC Application**
  - Palm OS
  - Networking Infrastructure
- **Communication** (With HHC Server)
- **User Interface**
- **Data Management**

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?

HHC Server

A simple interface: from within our HHC Server Architecture

Data server

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 \(m \cdot 10^n \div 32,768\), 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:
Employee {
  String name;
  String address;
  Integer year_of_birth;
  etc.
}

Interfaces

PARAMETERS

Compute Mean and Variance

INTERFACE

RETURNS

Implementation

Module A

Module B

Computation of key statistics

MEAN, VARIANCE

HIDDEN From Module A!!

- One module should not be concerned with other module's implementation
  - \(\rightarrow\) "Separation of concerns."
- One module should see the other only through its interface - implementation details hidden.
  - \(\rightarrow\) 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
Computes key statistics

Module B
Compute Mean and Variance

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

"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

Module A
Computes key statistics

Module B
Compute Mean and Variance

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 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

Hello: I'm the HHC of Airplane1234
Hello: I'm the gate 32 server
These were the unruly passengers on last flight "Passengers noted"
Tell me about the passengers of my next flight
Return Passenger Data
Tell me about the weather at my next destination.
Return Weather Data
(Might be passed as an array of a compound data type "passenger," which in turn is composed of standard types like integer, and string)
Another Interface 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
Internal functionality
Returns

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 (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**

![Diagram]

- authentication → failure
- choose objective → other objectives
- account → no accounts
- amount → balance exceeded!

**Goals**

More on layering

by

David G. Messerschmitt

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
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 as a server to the layer above

**Layering**

- Elaboration or specialization

  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>Internet Explorer</td>
<td></td>
</tr>
</tbody>
</table>

**Major layers**

- Network
- Operating system
- Middleware
- Components and frameworks
- Application
Data and information

Application
Deals with information
Assumes structure and interpretation

Infrastructure
Deals with data
Ignores structure and interpretation

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

Bob
US Postal Service
Shipping Container
ABC Airlines

Alice
UK Royal Mail
Envelope
Shipping Container

Example 2

Web server
Web page
Screen
Web browser
HTML

Application
File
Message
Message

Operating system
File system

Network
Fragmentation
Collection of packets
Assembly

Example 1

Bob sends a letter to Alice

Bob
US Postal Service
Shipping Container
ABC Airlines

Alice
UK Royal Mail
Envelope
Shipping Container

Example 2

Web server
Web page
Screen
Web browser
HTML

Application
File
Message
Message

Operating system
File system

Network
Fragmentation
Collection of packets
Assembly
Example 3

HHC Server

HHC Server Application
Passenger Information
message

HHC Client Application

Windows OS

Networking Infrastructure
(Contains: TCP/IP, WiFi)

Collection of Packets

Example 3: Network Infrastructure Expanded

HHC Server Application

Passenger Information
message

HHC Client Application

Windows OS

TCP transport layer

Packets

Networking Infrastructure
(Contains: TCP/IP, WiFi)

Collection of Packets

Networking Infrastructure
(Contains: TCP/IP, WiFi)

Example 4

HHC Server

Airline Dataserver

“Send me today’s flight information”

HHC Server Application

message

DBMS

Unix OS

Networking Infrastructure
Layers within TCP/IP, WiFi

Collection of Packets

Networking Infrastructure
Layers within 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