Class Announcements
Architecture

- How do you begin to architect a solution for a problem like this?
- Break it into modules!
- New in-flight seatback system
  - Sell upgrades and seat swaps
    - (People who want to get away from sick people ...)
  - More legroom
  - Offer to exchange seats
Architecture

Wireless Link

Seat back devices
  Wireless Link
  servers

HEADQUARTERS

Airline Dataserver
Seatback Architecture

When a module is composed of sub-modules, the architecture is **hierarchical**.
- We also make use of layers
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?
In-plane Server

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

Server Application
- LinuxOS
- Networking Infrastructure

Computation of key statistics
- Communication with airline database
- Communication with seat backs

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 Architecture

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

- HHC Application
- Linux
- Networking Infrastructure
- Communication with HHC
- Communication with airline database
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*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

INTERFACE

Computation of key statistics

RETURNS

Compute Mean and Variance

2 Numbers of float type that signify: Mean, Variance
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
Implementation

Compute Mean and Variance

Module A

Computation of key statistics

Module B

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.

\[
\begin{align*}
\text{SUM} & = \sum_{i=1}^{N} x_i \\
\text{MEAN} & = \frac{\text{SUM}}{N} \\
\text{VARIANCE} & = \sum_{i=1}^{N} (x_i - \text{MEAN})^2
\end{align*}
\]
Implementation

Module A

Computation of key statistics

Module B

Compute Mean and Variance

Implementation 1:

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

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

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

“\text{I need to get the sum, I’ll just take it from B}”

Should he use it?

\begin{itemize}
  \item NO!!!! Why??
\end{itemize}

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**.
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

Hello: I’m the HHC of Airplane#1234

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: authentication**

**Parameters**
- Identity (card in slot)
- Institution (card in slot)
- PIN (typed on keypad)

**Internally**, 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

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

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
Layer above is a client of the layer below

Layer below as a server to the layer above

Each layer provides services to the layer above....

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

Microsoft position

DOJ position

Application

Components and frameworks

Infrastructure
Major layers
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
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

Envelope

US Postal Service

Shipping Container

ABC Airlines

Alice

Envelope

UK Royal Mail

Shipping Container
Example 2

- Web server
- Web browser
- Web page
- Screen
- HTML
- File
- Message
- Application
- Operating system
- File system
- Network
- Fragmentation
- Collection of packets
- Assembly
Example 3

HHC Server Application → Passenger Information → HHC Client Application

Windows OS

Networking Infrastructure (Contains: TCP/IP, WiFi)

Palm OS

Networking Infrastructure (Contains: TCP/IP, WiFi)
Example 3: Network Infrastructure Expanded

- HHC Server Application
  - Windows OS
    - TCP transport layer
      - WiFi Link Layer
        - WiFi Physical Layer
  - Passenger Information
- HHC Client Application
  - Palm OS
    - TCP transport layer
      - WiFi Link Layer
        - WiFi Physical Layer

Networking Infrastructure

Message

Packets

Radio Signals
Example 4

HHC Server Application

Windows OS

Networking Infrastructure
Layers within TCP/IP, WiFi

“Send me today’s flight information”

DBMS

Unix OS

Networking Infrastructure
Layers within: TCP/IP, WiFi

HEADQUARTERS

Airline Dataserver
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