Announcements

- Database Lab available
  - You will need to attend one session to do the database assignment
  - Tutorial will also be available, run by TAs

- Assignment 3 posted later this week
  - DUE next Monday, 11/14

- For next time read:
  - Messerschmitt Ch. 7

Modularity and Layering

- The most important step to reduce/control 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 (e.g. frequent flyer points)
  - Flight attendant discriminates based on this
  - Free drinks, meals, and pillows to valued customers

Example Concept:

- 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

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

HHC Architecture

Each Tier is decomposed into modules

HHC Architecture

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

Some aspects of software complexity

1) The number of elements (or participants) increases → system’s complexity increases
2) The problem domain is complex
3) A lot of constraints
4) Every case must be foreseen
5) Continuous vs discrete, cannot 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
  - “Easier to code for one goal than ten goals”
- Interoperability
- Reusability

Student Talks

- Wai-Son Wong
A simple interface: from within our HHC Server Architecture

- **Computation of key statistics**
- **Compute Mean and Variance**

**Windows OS**

**Networking Infrastructure**

**Computation**

**Communication with HHC**

**Communication with airline database**

---

### Interfaces

- **PARAMETERS**: N numbers of float type
- **INTERFACE**: 2 Numbers of float type that signify:
  - Mean, Variance
- **RETURNS**: 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 A**

- **Computation of key statistics**

**Module B**

- **Compute Mean and Variance**

**Implementation 1**

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

- One module should not be concerned with other module's implementation
- One module should see the other only through its interface - implementation details hidden.
  - "Separation of concerns.
- Abstraction

**Implementation 2**

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

- 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 A**

- **Computation of key statistics**

**Module B**

- **Compute Mean and Variance**

**Implementation 2**

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

**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.
  - Modern languages allow developers to make this explicit (i.e., developer must affirmatively declare all publicly available items)
- Making a modules implementation details inaccessible to other modules is called **encapsulation**
Interfaces

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

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

Possible software interface
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
- Options (e.g., transaction record)

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

Returns
- Screen message
  - "Invalid PIN", or
  - Menu of available actions

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?
Protocol: cash_withdrawal

- authentication → failure
- choose objective → other objectives
- account → no account at this bank?!
- amount → balance exceeded!

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

US Postal Service

ABC Airlines

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
  - Ignores structure and interpretation

- Infrastructure
  - Deals with data

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

HHC Server Application

- Passenger Information
  - HHC Client Application
  - Windows OS
  - Networking Infrastructure (Contains: TCP/IP, WiFi)

Collection of Packets

Example 3: Network Infrastructure Expanded

- HHC Server Application
  - Passenger Information
    - Windows OS
    - Networking Infrastructure (Contains: TCP/IP, WiFi)

- HHC Client Application
  - Passenger Information
    - Palm OS
    - Networking Infrastructure (Contains: TCP/IP, WiFi)

TCP transport layer

WiFi Link Layer

WiFi Physical Layer

Radio Signals

---

---

---

---
Example 4

```
<table>
<thead>
<tr>
<th>HHC Server Application</th>
<th>“Send me today’s flight information”</th>
<th>DBMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows OS</td>
<td></td>
<td>Unix OS</td>
</tr>
<tr>
<td>Networking Infrastructure Layers within TCP/IP, WiFi</td>
<td>Collection of Packets</td>
<td>Networking Infrastructure Layers within TCP/IP, WiFi</td>
</tr>
</tbody>
</table>
```

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