Announcements

- 2nd Folio Article due today

Read
- Messerschmitt Ch 11.1 - 11.2 (325-335)
- Messerschmitt Ch 15.1 - 15.2 (415-425)
Student Talks
Architecture Example
Architecture

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

HEADQUARTERS

Airline Intranet

HHC Server

Wireless Link

HHC

Airline Datasever
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?
HHC Server Application

Windows OS

Networking Infrastructure

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

Computation of key statistics

Communication with HHC

Communication with airline database

Standard Database “queries” (SQL) relayed to DBMS via OS and infrastructure
Data server

DBMS

Database

Standard Database "queries" (SQL) from HHC Server

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

List of numbers

Computation of key statistics

Mean, Variance

Compute Mean and Variance

HHC Application
Palm OS
Networking Infrastructure

Communication with airline database

Communication with HHC
Interfaces

N numbers of Float type

Compute Mean and Variance

2 Numbers of float type that signify: Mean, Variance

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 $m*10^{n}/32768$, 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:
Employee {
    String name;
    String address;
    Integer year_of_birth;
    etc.
}
Interfaces

PARAMETERS

Computation of key statistics

N numbers of Float type

INTERFACE

Compute Mean and Variance

2 Numbers of float type that signify:
Mean, Variance

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

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

Module B
Compute Mean and Variance

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

\text{Implementation 1:}

**Should he use it?**

\text{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**.
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

(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)

- Click to edit Master text styles
  - Second level
  - Third level
  - Fourth level
  - Fifth level

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

Money output slot
  • Returns money
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
- 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!

• Click to edit Master text styles
  • Second level
  • Third level