Architecture Example

Example Concept:

- 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.

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

- Again, we use layering and hierarchy
- Between each module, we specify an interface

Data server

- Standard Database "query" (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 our HHC Server Architecture

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

Interfaces

- Compute Mean and Variance
- Computation of key statistics
- N numbers of Float type
- 2 Numbers of float type that signifies
- 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 $2^{16} = 65,536$
- Float
  - "number of the form $m \times 10^y$, where $m$ is in the range 32,767 to 32,768 and $n$ is in the range -255 to +255"
  - 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.)

15
16
17
18
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 module’s implementation details inaccessible to other modules is called **encapsulation**.
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 is a 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 unlucky passengers on last night's 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 return 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

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

- 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

Major layers

- Network
- Operating system
- Middleware
- 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
Envelope
US Postal Service
Shipping Container
ABC Airlines

Alice
Envelope
UK Royal Mail
Shipping Container

Example 2

Application
- Web server
- Web page

Operating system
- File
- Message

Screen
- Web browser
- HTML

Network
- File system
- Collection of packets
- Assembly

Example 3

HHC Server
- Passenger Information
- HHC Client Application
- Windows OS
- TCP transport layer
- WIFI Link Layer
- WIFI Physical Layer
- Networking Infrastructure

Example 3: Network Infrastructure Expanded

HHC Server Application
- Passenger Information
- HHC Client Application
- Windows OS
- TCP transport layer
- WIFI Link Layer
- WIFI Physical Layer
- Networking Infrastructure

Example 4

HHC Server
- Flight Information
- DIMMS
- Network Infrastructure
- 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