ISM 50 - Business Information Systems

Lecture 13

Instructor: John Musacchio
UC Santa Cruz
November 8, 2007

Announcements

- 2nd Folio Article
  - Due Tuesday
- Assignment 4 posted
  - DUE November 15.
- Read
  - Messerschmitt Ch 11.1 - 11.2 (325-335)
  - Messerschmitt Ch 15.1 - 15.2 (415-425)
- Presenters for Tuesday
  - Laura Maldonado (Business paper)
  - Blain Hockridge (Business paper)

Student Talks

Xinyu Zhu (Business paper)
Jenny Le (Business paper)

Architecture

- 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

Example Concept:

Architecture

- 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?
A simple interface: from within our HHC Server Architecture

Computation
Mean, Variance
List of numbers

HHC Application

Compute Mean and Variance

Communicate with IHC

Communication with airline database

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

Integer
- "natural number between -32,767 and +32,768"
- Could be represented (in many ways) by 16 bits
  - since $2^{15} = 65,536$

Float
- "number of the form $m \times 10^n / 32,768$, 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 $7 \times n$ bits

Example data types

Example: Employee {
  String name;
  String address;
  Integer year_of_birth;
  etc.
}

Compound data types

Programmer-defined composition of basic data types
**Interfaces**

PARAMETERS

N numbers of Float type

2 Numbers of float type that signify: Mean, Variance

INTERFACE

Compute Mean and Variance

RETURNS

**Computation**

N numbers of Float type

2 Numbers of float type that signify:
Compute Mean and Variance

**PARAMETERS**

Module B

Compute Mean and Variance

**INTERFACE**

RETURNS

**Implementation**

Module A

Computation of key statistics

Module B

HIDDEN From Module A!

- 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.

**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**.

- 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 \(\equiv\) 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

HHC Server

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

Another Interface Example:
Automatic teller machine (ATM)

What is the interface between this machine and the customer?

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

Steps

Define available actions
Define, for each higher level function, a protocol
- Single action or a finite sequence of actions
<table>
<thead>
<tr>
<th>Action: authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Parameters</strong></td>
</tr>
<tr>
<td>- Identity (card in slot)</td>
</tr>
<tr>
<td>- Institution (card in slot)</td>
</tr>
<tr>
<td>- PIN (typed on keypad)</td>
</tr>
<tr>
<td>Internally, it contacts institution and matches against its database, institution noted for all subsequent actions (example of state)</td>
</tr>
<tr>
<td><strong>Returns</strong></td>
</tr>
<tr>
<td>- Screen message (&quot;Invalid PIN&quot; or menu of available actions)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Action: specify_account</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Parameters</strong></td>
</tr>
<tr>
<td>- Account (touch screen from menu of choices)</td>
</tr>
<tr>
<td>Internally, choice noted for all subsequent actions (another example of state)</td>
</tr>
<tr>
<td><strong>Returns</strong></td>
</tr>
<tr>
<td>- None</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Action: amount</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Parameters</strong></td>
</tr>
<tr>
<td>- Dollars_and_cents (typed on keypad)</td>
</tr>
<tr>
<td>Internally, amount noted (another example of state)</td>
</tr>
<tr>
<td><strong>Returns</strong></td>
</tr>
<tr>
<td>- Success or failure (state dependent, for example for a withdraw failure when dollars_and_cents exceeds balance)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Protocol: cash_withdrawal</th>
</tr>
</thead>
<tbody>
<tr>
<td>What is the sequence of actions?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Protocol: cash_withdrawal</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Diagram of cash withdrawal protocol]</td>
</tr>
</tbody>
</table>

More on layering

by David G. Messerschmitt
Example 1

Bob sends a letter to Alice

US Postal Service
Shipping Container

UK Royal Mail
Shipping Container

ABC Airlines

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 as a server to the layer above

Three types of software

Application
- Components and frameworks:
  - What is in common among applications
- Infrastructure:
  - Basic services (communication, storage, concurrency, presentation, etc.)

Major layers

Applications
- Application frameworks and components
  - Middleware
  - Operating system
  - Network

Data and information

Application
- Deals with information
  - Assumes structure and interpretation
  - Ignores structure and interpretation
- Infrastructure
  - Deals with data

Example 2

Web server
Web page
Screen
Web browser
HTML

Application
Operating system
File system

Network
Fragmentation
Collection of packets
Assembly
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

Retain the
- values
- order
- number of bits in a package

Example 3

Example 3: Network Infrastructure Expanded

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

At most, data types

**Data and information**

- Application
  - Deals with information
  - Assumes structure and interpretation
  - Assumes standard data types

- Infrastructure
  - Deals with data types

---

**Architecture**

- HHC Server
- Wireless Link
- HHC
- Airline Intranet
- Airline Dataserver
- HEADQUARTERS

---

**Components**

- Component: A subsystem purchased "as is" from an outside vendor
- (Alternative – building your own subsystem)

A component implementation is encapsulated (although often configurable)

**HHC Architecture**

- HHC Application
  - Coordination With HHC Server
  - User Interface
  - Data Management

The Palm OS we are buying "off the shelf" and integrating into our architecture. The Palm OS is a **component**.
Other Examples of components

- Computer
- Disk drive
- Network
- Network router
- Operating system
- Integrated circuit
- Database management system

Interoperability

- Components are interoperable when they interact properly to achieve some desired functionality.
- Increasingly component interoperability cannot be dependent on end-user integration.
  - PC and peripherals
  - Enterprise, inter-enterprise, consumer applications
  - Role for standardization

Outsourcing

- Outsourcing: A subsystem design is contracted to an outside vendor.
- Responsibility is delegated.

HHC Architecture

- Suppose we choose to pay another firm to develop the user interface.
- This is called Outsourcing.
- Why would we do this?

System Integration

- Suppose we bring together all these subsystems and test them.
- This is called System Integration.

System Integration

- Bring together subsystems; make them work together; to achieve a goal.
- Requires:
  - Testing
  - Making modifications to architecture and/or subsystem implementation.
Can System Integration be Outsourced?
- Of course!

Supplier Types
- Three types of suppliers:
  - Component Suppliers
  - Custom Subsystem Developers
  - System Integrators
- (Some suppliers are 2 or even 3 of above.)

Two ways to sell Software

<table>
<thead>
<tr>
<th>Product</th>
<th>Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer installed and operated</td>
<td>Functionality provided over a wide-area network</td>
</tr>
<tr>
<td>Often (but not necessarily) sold or licensed at a fixed price</td>
<td>Often (but not necessarily) sold by subscription</td>
</tr>
</tbody>
</table>

Recall: Infrastructure and Applications
- Infrastructure
  - Equipment and/or software used by many applications
- Applications
  - Provide specific capabilities and features serving individual users.

Four possibilities

<table>
<thead>
<tr>
<th>Product</th>
<th>Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft Office</td>
<td>Hotmail</td>
</tr>
<tr>
<td>Application Infrastructure</td>
<td>Internet DNS</td>
</tr>
</tbody>
</table>

Application Service Provider
- Two types
  - Bundled
    - An infrastructure provider bundles applications with their infrastructure
    - Example: AOL, telephony service providers
  - Unbundled
    - A provider of an application service without providing an infrastructure service
    - Examples?
Examples of unbundled ASP model

- Yahoo: Web-based calendar
- Hotmail: Web-based email
- Schwab: Web-based stock trading

Unbundled ASP model

Advantageous to user
- Proven way to reduce installation, integration, and maintenance costs
- Contractual obligation for availability and quality
- Location independence

Advantages to supplier
- Ongoing revenue stream supporting upgrade and maintenance
- Usage-based revenue better aligned with user's value proposition
- Opportunity for price discrimination, advertising revenue, etc.

Some pricing alternatives

Price discrimination?
Usage dependent?
Terms and conditions
- Fixed, leasing, per-use, subscription
- Warrantee, service level agreements

Bundles
- Maintenance, support, releases, provisioning and operations

Who pays?
- Sometimes not the end user

Infrastructure acquisition

Application acquisition
**Stovepipe vs. Integrated Infrastructure**

<table>
<thead>
<tr>
<th>Stovepipe Architecture</th>
<th>Integrated Infrastructure</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Turnkey Solution</strong></td>
<td></td>
</tr>
<tr>
<td>Single supplier provides all encompassing solution</td>
<td>Separate infrastructure that can support many applications</td>
</tr>
<tr>
<td>(complete with infrastructure)</td>
<td>(complete with infrastructure)</td>
</tr>
</tbody>
</table>

**From stovepipe to layering**

- Data
- Voice
- Video

Many applications

Integrated Infrastructure (Maybe broken into Additional layers.)

Application-dependent infrastructure

Application-independent

**Stovepipe vs. Integrated Infrastructure**

- What are some examples of each?

- What are the advantages of each approach?

**Vertical Integration vs. Diversification**

- A company is **vertically integrated** when it makes rather than buys the subsystems in its products.

- A **diversified** company produces products across different industry segments.