Announcements

- Database Assignment 2 to be posted soon
- Does Assignment 3 require more time?
- Assignment 4 on reading to be posted soon
- Pizza party next week
- Business paper due next week
- Feedback when?

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

- We also make use of layers

HHC Architecture

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?

In-plane Server

![In-plane Server Diagram]

Data server

![Data server Diagram]

Interfaces

![Interfaces Diagram]

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 +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.
}

Interfaces

Parameters
- N numbers of Float type
- Computation of key statistics
- INTERFACE
- RETURNS

Compute Mean and Variance

Implementation

Module A
- Computation of key statistics
- Implementation 1:
  - $\text{MEAN} = \frac{\sum_{i=1}^{N} x_i}{N}$
  - $\text{VARIANCE} = \sum_{i=1}^{N} (x_i - \text{MEAN})^2$
- 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

Module B
- Computation of key statistics
- Implementation 2:
  - $\text{SUM} = \sum_{i=1}^{N} x_i$
  - $\text{MEAN} = \frac{\sum_{i=1}^{N} x_i}{N}$
  - $\text{VARIANCE} = \sum_{i=1}^{N} (x_i - \text{MEAN})^2$
- 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.
Implementation

Module B

Compute Mean and Variance

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

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

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

Should he use it?

- NO!!! Why??
- Either A should compute “SUM” himself, or sit down with B and redesign the

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**.

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 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

Another Interface Example:

**Automatic teller machine (ATM)**

What is the interface between this machine and the customer?
<table>
<thead>
<tr>
<th><strong>Steps</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Define available actions</td>
</tr>
<tr>
<td>Define, for each higher level function, a protocol</td>
</tr>
<tr>
<td>- Single action or a finite sequence of actions</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Interface building blocks</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Message on screen</td>
</tr>
<tr>
<td>Keypad</td>
</tr>
<tr>
<td>Card reader</td>
</tr>
<tr>
<td>Money output slot</td>
</tr>
<tr>
<td>Printer</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Action: authentication</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters</td>
</tr>
<tr>
<td>Internal functionality</td>
</tr>
<tr>
<td>Returns</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Action: authentication</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters</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>Returns</td>
</tr>
<tr>
<td>- Screen message (&quot;Invalid PIN&quot; or menu of available actions)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Action: specify_account</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters</td>
</tr>
<tr>
<td>Internal functionality</td>
</tr>
<tr>
<td>Returns</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Action: specify_account</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters</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>Returns</td>
</tr>
<tr>
<td>- None</td>
</tr>
</tbody>
</table>
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

Layering builds capability incrementally by adding to what exists.

Elaboration or specialization

Existing 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.

"Package" = file, message

Infrastructure deals with a "package" of data (non-standard terminology):
- collection of bits
- specified number and ordering

infrastructure stores and/or communicates "packages" while maintaining data integrity

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 browser

**Operating system**
- File system

**Network**
- Collection of packets
- Assembly

### Example 3: Network Infrastructure Expanded

- Seatback Application
  - Passenger Information
  - TCP transport layer
  - Linux OS
  - Packets

- Airplane Server
  - Linux OS
  - TCP transport layer
  - WiFi Link Layer

- Passenger Information
  - WiFi Physical Layer
  - Radio Signals

### Computer & Comm. Industry Structure

**Two ways to design a system**

1. **System requirements**
2. **Requirements**
3. **Decomposition from system requirements**
4. **Assembly from available components**

### Components

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

- A component implementation is encapsulated (although often configurable)

### Seatback Architecture

- **HHC Application**
  - Linux OS

- **Networking Infrastructure**

  **User Interface**
  - Data Management
  - Coordination

  **Server**
  - With Plane

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

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

Why is a component implementation encapsulated?

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</td>
<td></td>
</tr>
<tr>
<td>Infrastructure</td>
<td>Internet DNS</td>
</tr>
<tr>
<td>Personal computer</td>
<td></td>
</tr>
</tbody>
</table>

Application Service Provider

- Two types
  - Bundled
    - An infrastructure provider bundles applications with their infrastructure
      - Example: Comcast, 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
- Gmail: 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

Unbundled ASP model (con’t)

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
- warranty, service level agreements
Bundles
- maintenance, support, releases, provisioning and operations
Who pays?
- sometimes not the end user

Infrastructure acquisition

<table>
<thead>
<tr>
<th>Infrastructure</th>
<th>Build and operate</th>
<th>Build but do not operate</th>
<th>Do not build but operate</th>
<th>Neither</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trend</td>
<td>Outsourced operations</td>
<td>System integration</td>
<td>Service provider</td>
<td></td>
</tr>
</tbody>
</table>

Application acquisition

<table>
<thead>
<tr>
<th>Application</th>
<th>Develop internally</th>
<th>Buy as product</th>
<th>Contract development</th>
<th>Product w/ customization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trend</td>
<td>Software supplier</td>
<td>Outsource developer</td>
<td>Supplier consultants</td>
<td></td>
</tr>
</tbody>
</table>
Stovepipe vs. Integrated Infrastructure

- **Stovepipe architecture**
  - Turnkey Solution
    - Single supplier provides all encompassing solution
    - (complete with infrastructure)

- **Integrated Infrastructure**
  - Separate infrastructure that can support many applications

---

From stovepipe to layering

- Data
- Voice
- Video

Many applications

Integrated Infrastructure
(Maybe broken into Additional layers.)

Application-dependent infrastructure

Application-independent

---

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.

---

Vertical Integration vs. Diversification

- Why do customers favor less vertical integration?
  - Prefer competition amongst component suppliers
  - Mix and match components
  - Reduced lock in
  - **Disadvantages??**
    - Customer needs to integrate components from different suppliers.

---

Vertical Integration vs. Diversification

- Why do customers favor diversification?
  - Reduce coordination costs by having to deal with fewer suppliers.

---

Stovepipe vs. Integrated Infrastructure

- What are some examples of each?

- What are the advantages of each approach?
General Trend

- Less Vertical Integration
- More Diversification
- Of course there are exceptions...

今日の供給体制

今日の供給体制

- Applications
- Frameworks and components
- Middleware
- Infrastructure (network, OS) software
- Equipment (network, computers)
- Semiconductors, components

Role of Venture Capital in Computing.

- Open interfaces allow small firms to contribute components without having to develop entire solution
- Fast decision making and no supplier lock-in.
- Other Advantages?

Standardization

Scope of a standard

Included:
- interfaces (physical, electrical, information)
- architecture (reference model)
- formats and protocols (FAP)
- compliance tests (or process)

Excluded:
- implementation
- (possibly) extensions
Reference model

Decide decomposition of system
- where interfaces fall
Defines the boundaries of competition and ultimately industrial organization
- competition on the same side of an interface
- complementary suppliers on different sides
- hierarchical decomposition at the option of suppliers
- (possibly) optional extensions at option of suppliers

Some issues

Once a standard is set
- becomes possible source of industry lock-in; overcoming that standard requires a major (~10x?) advance
- may lock out some innovation
In recognition, some standards evolve
- IETF, CCITT (modems), MPEG
- backward compatibility

Types of standards

de jure
- Sanctioned and actively promoted by some organization with jurisdiction, or by government

de facto
- Dominant solution arising out of the market
- Voluntary industry standards body
Industry consortium
Common or best practice
Examples?

Examples

de jure
- GSM, ISDN Telephone interface

de facto
- Microsoft Windows API (Application Programming Interface)
- Intel Pentium instruction set
- Voluntary industry standards body
- IEEE (Institute of Electrical and Electronic Engineers)
- IETF (Internet Engineering Task Force)
Industry consortium
- bluray
Best practice
- Windowed GUI

The changing process

- As technology and industry move more quickly, the global consensus standards activity has proven too unwieldy
  - e.g. ISO
- "New age" standards activities are more informal, less consensus driven, a little less political, more strategic, smaller groups
  - e.g. IETF
Programmable/extensible approaches for flexibility
  - e.g. XML, Java

Old giving way to the new

Old giving way to the new
Reasons for change

- From government sanction/ownership to market forces
  - Increasing fragmentation
  - Importance of time to market
- Greater complexity
  - Less physical/performance constraint for either hardware or software

Lock-in

- (Particularly open) standards reduce consumer lock-in
  - Consumers can mix and match complementary products
- Increase supplier lock-in
  - Innovation limited by backward compatibility
  - e.g. IP/TCP, x86, Hayes command set

Aside: Network Effects

- The value of owning some products goes up if lots of other people have it too.
  - Examples?
- This phenomenon is called “network effects”
- How do standards influence network effects?

Network effects

- Standards can harness network effects to the industry advantage
  - Revenue = (market size) x (market share)
  - Increases value to customer
  - Increases competition
    - Only within confines of the standard
    - But forces customer integration or services of a system integrator

Why standards?

- de jure are customer driven to reduce confusion and cost
- de facto standards are sometimes the result of positive feedback in network effects
- Customers and suppliers like them because they
  - increase value
  - reduce lock in
- Governments like them because they
  - promote competition in some circumstances
  - May believe they can be used to national advantage

Approaches

- Consensus
- ISO
- Collaborative design
- MPEG
- Competitive “bake off”
- IETF
Open vs. Proprietary Standards

- Open standard – a standard that is well documented, unencumbered by intellectual property rights and restrictions, and available to any vendor.

- What are the advantages?

- What are the disadvantages?

Why companies participate

- Pool expertise in collaborative design
  - e.g. MPEG

- Have influence on the standard

- Get technology into the standard
  - Proprietary, with expectation of royalties
  - Non-proprietary

- Reduced time to market

Standards applied to Business Processes?

- Can you standardize business processes?

- Yes:
  - ISO 9000
    - A set of standardized business processes for Quality Management
    - Supports TQM (Total Quality Management)
  - RosettaNet
    - A set of standardized business processes, and accompanying standardized data interfaces/formats for conducting e-business.