Announcements

- 2nd Folio Article due today
- Assignment 4 out, (due Wednesday 5/20)

For Wednesday, read:
- Messerschmitt Ch 11.1 - 11.2 (325-335)
- Messerschmitt Ch 15.1 - 15.2 (415-425)

Presenters for Wednesday 5/13
- Matthew Villarreal
- Xuanyu Pang

Midterms back on Wednesday

Today's Class

- Review of midterm
- Database project overview
- Student presentations
- Business paper proposals and research
- Architecture: Modularity and Layering
- Computer and Communications Industries

Database Assignment

-- see assignment, posted
-- TA Lei Yang has signups for lab sessions to help you learn MS Access and complete the project. There's room for about 40 students per session, so we will have signup sheets at Wed. lecture for each of 4-5 sessions, depending on demand.
-- You may be able to read MS Access help and do the lab on your own.
-- Come early Wed. to sign up for your first choice of lab times. We may offer additional labs if there is demand for them, let us know.

Student Presentations

Larry Tse
Julian Chytrowski
Business paper proposals

- Characterize our HBS/Stanford case studies:
  - Frito-Lay, Cisco: companies invest in ERP systems to gain a competitive advantage in traditional markets
  - Alibris: a business designed from scratch around web technology. They invest in IT to build an ecommerce business as an intermediary
  - Sun, MySQL: IT supplier companies that build systems to make their customers more competitive. As a supplier, their own competitiveness depends on the price/performance of their products

- Like Frito-Lay, Cisco: investment in ERP
  - Best Buy, Best Western, Christian Dior
  - Coca-Cola, Costco, Dell
  - Exxon Mobile, Fedex, Hampton Fitness
  - Joie de Vivre, Mary Kay, McDonalds
  - Nike, Plantronics, 7-Eleven
  - Siemens, Southwest Air, Toyota
  - Verizon, Walmart, Walt Disney

- Like Alibris: web-focused; or an ecommerce intermediary business model seeking a network effect
  - Amazon, Apple?, Borders?
  - Charles Schwab, Citibank?, Craigslist
  - eBay, Google, Netflix
  - Overstock.com, Skype, Wells Fargo?
  - Wikipedia

Business proposal feedback: citing sources

- No plagiarism:
  - More than thirty words verbatim must be cited.
  - Any facts or figures that are not your own must be cited in the body of the text.
  - Sun’s servers ranged in price from $14,000 to over $1 million and had an estimated 65% gross margin, while the company’s workstations were priced at an average of $15,000 and had an estimated gross margin of only 38 percent [1].

References

[2] ...

Citing Sources: “print citations”

We are looking for five or more print-style citations among all the citations in the References section at the end of your paper. (A print source can also be published on the web.)

“Semiconductors have found a place in virtually every electronic device in existence. This helps explain why the industry was able to reach $200 billion in sales before a slump brought the figure back down in 2001. [1]. By 2003, chip sales rebounded and actually exceeded $200 billion [2].

References


Not a print citation
Citing Sources

- Use Wikipedia as an encyclopedia: it is a pointer to sources you cite, not a source itself. We'll accept one or two Wikipedia citations out of hopefully 15 - 25+ citations in the References section containing your end notes.
- The easiest way to lose points on your paper is to not cite sources!
- A guide is posted on the class website to help you cite your sources correctly:
  
  http://www.soe.ucsc.edu/classes/ism050/Spring09/cite.html

Business paper research (1)

- Company Website
- The company's own website is a reasonable place to start.
  
  Example: Walgreen's Drugs. www.walgreens.com
  
  Company information --> Investor relations.

Business paper research ideas (2)

- UCSC Library web resources
  
  http://library.ucsc.edu/research.html
  
  http://library.ucsc.edu --> Research --> How to Find Articles --> Business --> Company
  
  http://library.ucsc.edu/collect/businessweb.html#Company
  
  LexisNexis tutorial: Zimmerman's company research guide
  
  LexisNexis start page:
  
  
  Specific articles: http://library.ucsc.edu/ --> Online Journals List
  
  Economist, Wall Street Journal, Wired,...

Business paper research ideas (3)

- Private Company Research
- Business Week Research Tools: Companies and Industry Sector
  
  http://investing.businessweek.com
  
  http://en.wikipedia.org/wiki/Walgreens
  
  UNC Libraries references:
  
  http://www.lib.unc.edu/subjectguides/private_company/rankings.html
  
  Library of Congress
  
  http://www.loc.gov/rr/business/company/private.html

Business paper research ideas (4)

- Google Scholar search for "Walgreens drugs"
  
  Note Ranking Order: Porter article! & Walgreen's case study. (citations & salience)
  
  "Walgreen's Drug Dilemma" from Forbes. Use "UC-eLinks" OR online journals.
  
  Try "Recent Articles", set to 2008:
  
  Or, add "electronic prescriptions" to see articles including this topic.

Text, Chapter 6: Modularity and Layering

6.1. Software complexity
6.2. Modularity
6.3. Software layering
6.4. Good architectures, based on layering
Modularity: Implements Ch. 4
Architectural Design Principles

“Getting the architecture right is the most important step in the conceptualization and design of any system because it is the hardest aspect to change and it influences everything that follows.

The architecture is also the frontline defense against complexity, which is the most important limitation to what can be accomplished in software.”

Let’s boil Ch. 6 down to 14 terms you’ll need to understand:
- decomposition
- granularity
- hierarchy
- action
- interface
- data type
- protocol
- layering
- fragmentation
- data integrity
- horizontal structure
- spanning layer
- abstraction
- encapsulation

Fun example: a modular 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!

Architecture

- [Diagram showing HHC Server, Airline Intranet, Airline Dataserver, HEADQUARTERS, and Wireless Link connected to HHC]
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**.
  (Coarse, or fine.)
- Which is easier to debug and verify?
- Which is easier to define in an architecture?
  - Is the difference between microeconomics and macroeconomics primarily one of granularity, or of abstraction?

A simple interface: from within our HHC Server Architecture

Compute Mean and Variance

List of numbers

Computations of key statistics

Mean, Variance

N numbers of Float type

Interfacing with DBMS

Interfaces

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.
Interfaces

**PARAMETERS**

N numbers of float type

**Compute Mean and Variance**

Computation of key statistics

**INTERFACE**

RETURNS

Implementation

Module B

Compute Mean and Variance

Implementation 1:

\[ X_i, i=1..N \]

MEAN, VARIANCE

HIDDEN From Module A!

One module should not be concerned with other module’s implementation

\[ \rightarrow \] "Separation of concerns."

One module should see the other only through its interface - implementation details hidden.

\[ \rightarrow \] Abstraction

Implementation

Module B

Compute Mean and Variance

Implementation 2:

\[ X_i, i=1..N \]

MEAN, VARIANCE

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.

\[ \rightarrow \] Example: B does not declare "SUM" as a global variable.

Making a modules implementation details inaccessible to other modules is called **encapsulation.**
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

Those 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)**

What is the interface between this machine and the customer?

**Protocol: cash_withdrawal**

- authentication → failure
- choose objective → other objectives
- account → no accounts
- amount → balance exceeded!

**Horizontal structure, and spanning layers.**

Can you think of a company that had a dramatic recent market success through creating a spanning layer?

**Fragmentation example: serving a web page requested by a client. (How many tiers?)**

- Web server → Web page
- Web browser → Screen
- HTML

Application → Web page
- File system
- Network
- Collection of packets
- Assembly

Operating system → File system
- Network
- Fragmentation
Data integrity

Retain the
- values
- order
- number
of bits in a packet.

Requires work by specially designed hardware and software protocols to prevent corruption of packets in the presence of problems like electronic noise, network congestion, security breaches, etc.

Example: layering, interfaces, protocols, fragmentation, data integrity

Example 3: Network Infrastructure Expanded

Note: WiFi link layer protocol resends packets if they are dropped. TCP transport layer protocol resends groups of packets if their successful reception is not acknowledged.

Terms to know from Ch. 6 for final exam: (1)

- 6.1. Complexity—none, we already covered it
- 6.2. Modularity (p. 162+)
  - decomposition: most important step in system design
  - granularity: number of modules and range of functionality of each one. Coarse or fine
  - hierarchy: modules are themselves composed of smaller modules
  - action: one out of a menu of functions that a module will perform; invoked by a calling module
  - interface: a kind of contract between one module and another, specifying how they interact with each other
  - data type: a way for modules to agree on the structure and interpretation of data. (Don’t have to remember specific data types: integer, char, etc.)
  - protocol: a sequence of actions to agree on the structure and interpretation of data. (Don’t have to remember specific data types: integer, char, etc.)

Terms to know from Ch. 6 for final exam: (2)

- 6.3. Software layering (p. 172+)
  - layering: a type of decomposition, where higher layers are clients of lower layers/lower layers are servers for higher layers. Same as Ch. 4 definition (p. 120)
  - layers in a computing infrastructure: diagram on page 175, Fig. 6.7., one app. talking to another: Application 1 – middleware – OS – network – OS – middleware – application 2
  - fragmentation: a web page is fragmented into separate packets to be sent from a web server to a client browser (page 179, Fig. 6.8)
  - data integrity: see sidebar discussion, page 183.
  - horizontal structure: Fig. 6.13, page 186. Just means heterogeneity within a layer, causing extra complexity (example: multiple operating systems)
  - spanning layer: a technology layer that eliminates horizontal structure (the internet protocol, for example).

Terms to know from Ch. 6 for final exam: (3)

- 6.4. Good architectures through layering (p. 187+)
  - abstraction: hiding unnecessary detail.
  - Example: a good module interface hides all the complex stuff going on inside a module from the outside world. The outside world sees the module as an abstraction, through a simplified interface (contract). Key to success: choose appropriate abstractions!
  - encapsulation: making sure that complex internal details are invisible, and inaccessible, at a module’s interface.
Text, Chapter 7: Computer and Communications Industries

...this is closely related to our discussion of Chapters 4, 5, and 6: these industries have shaped themselves around the architectural concepts we’ve been talking about.

Two ways to design a system
- System requirements
- Requirements
  - Decomposition from system requirements
  - Assembly from available components

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

HHC Architecture
- HHC Application
  - Palm OS
  - Networking Infrastructure

Other Examples of components
- Computer
- Disk drive
- Network
- Network router
- Operating system
- Integrated circuit
- Database management system

The Palm OS we are buying “off the shelf” and integrating into our architecture. The Palm OS is a component.
**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</td>
<td>Functionality provided over a wide-area network</td>
</tr>
<tr>
<td>operated</td>
<td>Often (but not necessarily) sold or licensed at a fixed price</td>
</tr>
<tr>
<td></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>Infrastructure</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft Office</td>
<td>Hotmail</td>
</tr>
<tr>
<td>Personal computer</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

**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
- warrantee, service level agreements

**Bundles**
- maintenance, support, releases, provisioning and operations

**Who pays?**
- sometimes not the end user

**Infrastructure acquisition**

Infrastructure

- Build and operate
- Build but do not operate
- Do not build but operate
- Neither

Trend

Outsourced operations
System integrator
Service provider

**Application acquisition**

Application

- Develop internally
- Buy as product
- Contract development
- Product w/ customization

Trend

Software supplier
Outsource developer
Supplier, consultants

**Stovepipe vs. Integrated Infrastructure**

**Stovepipe architecture**

--- or ---

**Integrated Infrastructure**

- Separate infrastructure that can support many applications
- (complete with infrastructure)
From stovepipe to layering

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.

- 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.

General Trend

- **Less Vertical Integration**
- **More Diversification**
- Of course there are exceptions...
Today’s supplier structure

- 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

- Allow products or services from different suppliers or providers to be interoperable

Purpose of a standard?

- 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

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
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
- Hayes command set, Windows API, Pentium instruction set, Ethernet
- Voluntary industry standards body
  - OMG/CORBA, IAB/IETF, IEEE
  - Industry consortium
  - W3C/XML, SET
- 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. OMG, IETF, ATM Forum, WAP

Programmable/extensible approaches for flexibility
- e.g. XML, Java

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

Network effects

Standards can harness network effects to the industry advantage
- Revenue = (market size) * (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

Coordination of vendors
- OMG

Open 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.