Announcements

- Final Exam March 17, 2010
  - E2 180 (this room)
  - 8 AM -11 AM
- Final Exam Study Guide Posted on Website
- Business Paper Submission Due Today
- Assignment 4 submission due today
- Review Session by TA
  - Fri Mar 12, 2010, 9:30 AM - 10:40 AM
  - Jack Baskin 152

Student talks

Chapter 9

- Applications and the Organization

Build vs. Buy?

- Purchase off the Shelf
  - less time and cost
  - benefits of using a “standard” solution
  - support available
  - must mold org to app
  - no potential for competitive advantage
- Outsource
  - developers not as familiar with org as you
  - more opportunity for customizing than off the shelf
  - contractor may share knowledge with competitors
  - contractor may have too much bargaining power
- Make
  - most customizable of 3
  - easier iteration between conceptualization and development needed
  - most risky
  - org may lack competency to do it

Application Lifecycle

- It is important to think beyond acquiring an application
  - How do we come with the idea?
  - How do we architect it?
  - How do we implement?
  - How do we extend and maintain it?
- For this reason, the software engineering community came up with:
  - Application Lifecycle Model
Application Lifecycle

Stages:
1. Conceptualization
2. Analysis
3. Architecture Design
4. Development Evolution
5. Testing and Evaluation
6. Deployment
7. Operations, Maintenance, and Upgrade

1) Conceptualization

What is the vision?
- What are the objectives?
- What is the business case?

EXAMPLE: HHC to inform flight attendants which passengers are low and high value.

Business Case:
- Increase repeat business from high value customers.

1) Conceptualization -- Example:

2) Analysis

- Describe what the application will do.
- Don't make highly detailed specifications
- Describe scenarios in which it is used
  - (Use Cases)

2) Analysis -- Example

- Example: Scenario:
  - Premium Passenger Identification function
    - FA wants to see who are premium passengers to offer them a free drink
    - FA clicks "on" premium passenger radio button
    - HHC displays color-coded seat map.

3) Architecture Design

- Decompose the application into subsystems
  - Hardware, software
  - Try use commercial off the shelf subsystems
  - Try to use standard infrastructure layers
    - Operating system, network, middleware, etc.
3) Architecture Continued

- Define the functionality, interaction and interfaces of subsystems
- While doing this, consider
  - Scalability
  - Can we increase number of users easily?
  - Extensibility
  - Ability to add new features later
  - Administration
    - Is it hard to keep it working?

4) Development Evolution

- Develop the details
  - Develop/program custom subsystems
  - Have contractor build outsourced pieces
  - Put together with off-the-shelf components

- Incremental
  - Start with simplest implementation and get it working
  - Later add more features.

5) Testing

- A must!
- If architected well, we can test subsystems independently.
- Alpha test
  - Off-line test of prototype
- Beta test
  - Test in intended environment with cooperative users

6) Deployment

- Convert from previous processes if necessary
  - Example: CISCO ERP (all at once)
  - Or, you could do incrementally
- Train users
  - Example: Frito-Lay HHNC
- Data importation
  - (if necessary)

7) Operations, Maintenance, Upgrade

- Maintain Security
- Repair Problems
- Correct performance short comings (Cisco ERP)
- Add features
Application Lifecycle Model concluding remarks

- ALM rarely followed precisely
- Many times projects loop between stages
- ALM followed more closely in larger companies
- Alternative:
  - Rapid Iterative Prototyping
  - (Cisco did some of this in the ERP case.)

POP QUIZ!!

1) An action available at an object’s interface is a
   a) field   b) method  c) query  d) packet

2) Objects that are of the same class of objects are called
   a) instances of the class
   b) replicates of the class
   c) copies of the class
   d) regenerations of the class

3) A numerical value or data belonging to an object that is externally visible, and may be changeable is called
   a) an instance  b) a frame  c) a method  d) an attribute

Application Architecture

- Decomposition – Divide the architecture into interacting modules.

- Assembly- Find subsystems available for purchase

- Most architecture design is a mixture

Decomposition Example

- Example: manage bank accounts

- Decompose into software modules for
  - transaction processing,
  - statement generation

- Further decompose transaction processing module into deposit and withdraw modules...

Assembly Example

- example - ecommerce platform
  - Acquire
    - Linux pc (application server)
    - IBM Mainframe (data server)
    - Oracle DBMS
    - Apache Web Server Software
  - Assemble all pieces together,
  - Mix with custom developed application logic module.
Object-Oriented Architectures

- Object-Oriented Programming (OOP) Languages
  - C++
  - Java
  - Smalltalk
- The basic unit of modularity in OOP is an object.

Objects

- Example: Bank account
  - has a balance of $5000
  - belongs to Joe Schmoe
  - Is a checking account
- can have money deposited to it
- can have money withdrawn from it

Objects

- An attribute is a numerical value or data that is externally visible, and may be changeable.
  - Ex: The bank account’s balance is $5000
- A method is an action available at the object interface
  - Other objects invoke method, pass parameters and get returned data or other objects.
  - We can invoke the "check_balance" method and get returned the number $5000

Object Classes and Instances

- Some objects share types of attributes and methods.
  - They have the same class
- Example
  - Class: Bank_Account
  - Instances:
    - Schmoe_Acccount balance: $5000
    - Smith_Acccount balance: $10000
  - Each instance is a separate object with its own data

Declaring Classes

When we program, we define or "declare" each class we plan to use.
- Example: We plan to use a class called "bank_account"
- It will have the attributes: balance, owner, etc., ...
- It will have the methods: check_bal, withdraw, deposit, ...
  - Later on we fill in the details of what each method does.
- Once we declare a class, we can create instances of it.
  - Schmoe_account, smith_account, etc..

Method Invocation

- Objects communicate with each other by invoking each other’s methods
  - (method invocation)

  ATM Object
  - invoke: check_bal() returns $5000
  - invoke: withdraw( $500) return "successful"

  Schmoe_Account
  - Attributes: balance: $5000
  - Methods: withdraw( ) check_bal( )

- Terminology:
  - Client object -- object invoking the method is the
  - Server object - object whose methods are being invoked
Software Objects

- In OOP an object can
  - Represent a real world entity
    - Bank account
  - Be a proxy of a real world entity
    - Proxy of a customer
    - Other software talks to proxy using method invocations
  - Model a real-world entity
    - For purposes of simulation
    - Motion of a train

ORDBMS

- Earlier in the class we talked about relational DBMS
  - The most common database management system that organizes data into tables.
- ORDBMS (Object Relational DBMS)
  - Retrieve and store object instance data in a relational database

Remote Method Invocation

- Sometimes we want to allow an object to invoke methods on an object located on another machine.
- This is called Remote Method Invocation (RMI)
- Doing this requires middleware called
  - Distributed Object Management (DOM)

Software Reuse

- Size and complexity of applications growing dramatically
- Reuse needed to cut costs
- Reuse is difficult. Why?
- OOP was developed in part to promote re-use

Software components

- Software components are reusable modules that can be bought from outside vendors.
- How is a component different from an object?
  - More importance on
    - Encapsulation
    - Well defined and documented interfaces

Component Assembly Tools

- Visual or integrated development environment (IDE)
  - MS Visual Studio
  - IBM Visual Age
  - Symantec Visual Café
- Scripting Assembly - Text based
  - TCL
  - Perl
  - JavaScript
Software Frameworks

- A preexisting architecture and library of components from a common vendor to help developers
- Enables reuse, and ensures component interoperability.
- Examples:
  - Sun J2EE/Java Beans
  - Microsoft .Net

American Airlines Case- Systems Operations and Control center (SOC)

- Flight Dispatching** focuses on:
  - Flight Path
  - Fuel Load
  - en route weather, problems
  - Each dispatcher assigned a geographic area
- Load Planning:
  - optimize loading of passengers and freight
  - consider runway length, weather, plane type, etc.
- Crew Scheduling:
  - Crews under strict regulations about amount of time can work
  - Certain crews can fly certain planes
  - seniority
  - positioning for future flights

Dispatch Automation Package

- Flight tracking application
  - View 1
    - List of all flights dispatcher responsible for
  - View 2
    - Dependencies of one flight on other flights.
- Message tracking
  - e-mail to flight crew

View 1:

View 2:
Flight Tracking Application

- **Facts and Figures:**
  - written in Smalltalk
  - 210 classes
  - 2000 methods
  - 160000 lines of assembler code
  - 150000 object instances in memory at all times

Flight Tracking Application

- **Good architecture allowed extensions later**
  - Feature to allow dispatcher to focus on very really late flights
  - Flight lock - stop flights to airport for bad weather
  - In flight fuel calculation
  - Geographical flight monitor

Flight Tracking Application

- **OOP + good architecture made 3 changes easier**
  - Changed how flight was referenced, major change to data model
    - (1.5 weeks)
  - Introduced File servers to cache FOS data
    - (1 day, 4 weeks test)
  - Developed message queuing monitor
    - (1 wk, test 3 wk)

Flight Tracking Application

- **Did AA follow Application Lifecycle Model?**
  - Fall 90 - Built as prototype as a way for someone to teach himself OOP
  - May 91 - OK to develop application
  - Work divided (one person in charge of each)
    - User interface
    - data model
    - data exchange with FOS
  - Nov 91 - production installation complete
  - 2 months testing
  - Extensions added Later

AA IT Architecture

- **Flight Tracking Application**
  - Fall 90 - Built as prototype as a way for someone to teach himself OOP
  - May 91 - OK to develop application
  - Work divided (one person in charge of each)
    - User interface
    - data model
    - data exchange with FOS
  - Nov 91 - production installation complete
  - 2 months testing
Final Review

- Comprehensive
- More emphasis on 2nd half of class.

IT History

What distinguished the Data Processing Era from the Micro-Era?

- What distinguished the Micro-Era from the Networking Era?
- Why did Data Processing Managers feel threatened in the Micro-Era?

Messerschmitt 2.3

- Data vs. Information
- Information Pull vs. Information Push

O’Brien Chapter 2

- What is Porter’s Competitive model?
  - 5 forces...
- What are the Porter Strategies
  - Primary – Cost Leadership, Differentiation
    - Why must a company do at least one of these?
  - Supporting – Alliance, Growth, Innovation
- Concepts:
  - Switching costs, barrier to entry,
  - business function, business process,
  - business process reengineering,
  - knowledge vs information,...

Messerschmitt Ch 3

- ERP
- Decision Support Systems
- Electronic commerce
  - Inter-enterprise B2B
  - Consumer B2C
  - Inter Consumer C2C
  - Examples of each?
- Supply Chain Management
- Mass Customization
- Electronic Data Interchange

Messerschmitt Ch 4

- Information represented as data
- Regeneration
- Software Layering
  - OS, Middleware, Application
- File System
- Database Management System
- Message, Packet,
  - internet, Internet, intranet, extranet
Cash Flow Analysis

- Net Present Value
  - What is it?
  - How do you Calculate it?
- Rate of Return

Client Server Computing

- What is a client-server architecture?
- What is a three-tier client server architecture?
- What is a peer to peer architecture?
- What is a thin client?

3-Tier Client Server Architecture in General

- Takes inputs from client
- Decides what to be done next
- Decides what shared data to access
- Manipulates it
- Processes shared data

- Accept instructions from user
- Make requests of server
- Display responses of server
- Support multiple applications with common data
- Protect critical data
- Decouple data administration and application administration

- Shared data

Modularity and Layering

- Modularity
  - Why is Modularity a good way to architect things?
  - Separation of concerns
  - Interoperability
  - Reusability
  - Is Layering the same as Modularity?
  - What is Hierarchy?
  - What are Interfaces?
    - External view of a subsystem, defines how other subsystems can interact with it. Often done using actions, parameters, and return values.
    - What is Abstraction?
      - Hide irrelevant detail at the interface of a subsystem
      - Helps manage complexity
    - What is Encapsulation?
      - Enforce abstraction, by making subsystem details inaccessible at the interface

Computers and Communication Industries

- Infrastructure vs Application
- Decomposition vs Assembly
- System Integration
  - What is it?
  - Can it be outsourced?
- stovepipe vs Integrated infrastructure
  - Examples? What are the tradeoffs of each approach?
- Standardization
  - Why are they needed?
  - Why do companies participate?
  - Can a reference model be part of a standard?
  - Should an implementation be part of a standard?
  - What is an open standard?
  - What are network effects?

Programming an Application

- What is an algorithm?
- What is a protocol?
- What is a name?
- What is an address?
  - Represents physical, or "topological" location of something. Example - wired phone number
- What is a reference?
  - Abstract representation of location of entity. Example - cell phone number
Data Sharing

- What is a DBMS?
- What is a relational database?
- SQL
- Markup Languages
  - What is the difference between HTML and XML?

DBMS Responsibilities

- Hide Changes in the Database hardware from the Application
- Standard operations on the data, including searches, such a search is called a query.
- Separate Database Management from Applications, so that many applications can access the same data.
- Security, Integrity, Backup, fault tolerance, etc.

Applications and the Organization

- What are the advantages and disadvantages of
  - Buying off the shelf
  - Outsourcing
  - Making yourself
- What are the 7 steps in the application lifecycle model?

Application Architecture

- What is Object Oriented Programming
  - What are methods?
  - What are attributes?
  - What are instances?
- What are software components?
- What are component assembly tools?
- What are software frameworks?

Networks

- What are the 7 OSI Layers?
- Ethernet belongs to which layer?
- What is the difference between a MAC address and an IP Address?
- What is the difference between Time Division Multiplexing and Statistical Multiplexing?
- How does routing work on the Internet?
- How is an Internet Packet sent over an Ethernet?
- What does TCP do?

Link and Network Layer Interaction

- MAC Header
- Ethernet Frame
- Payload
- IP Header
- IP Payload
- Router
- Ethernet Hub
- Host A
- Host B
- Host C
- 128.114.00.202
- Step MAC header off frame. Forward IP packet based on Routing table.
Frito-Lay

What were the main reasons why Frito Lay deployed the HHC?

What changes in marketing strategy did Frito-Lay believe the HHC data would help enable?

How might the HHC project change Frito-Lay's competitive position with its direct competitors, new entrants, and its customers?

Cisco

What are some of the actions that Cisco took that contributed to the successful deployment of ERP?

What mistakes did Cisco make?

What are the most important lessons that another company that wants to deploy ERP could learn from Cisco's experience?

Alibris

How and why did Alibris plan to change Interlac's revenue model?

How did Alibris plan to avoid becoming dis-intermediated?

Why did Alibris abandon Thunderstone, software, and why did it choose to switch Oracle?

What made Alibris IT challenge particularly difficult compared to what other ecommerce companies faced?

Sun Case

What are the drawbacks of having thin-client vs. a traditional fat-client?

What selfish reasons does Sun have for advocating a thin-client model? Why does Microsoft prefer maintaining the dominance of the fat-client model?

What is Java, and what advantages does it have over other languages?

Since the case was written, which OS has come to dominate the web-server market?

MySQL Case

Who are the three biggest suppliers of DBMS? What competitive advantages does MySQL have in the Web Site data segment?

Why would large enterprises prefer database software from one of the three major DBMS providers, over MySQL's cheaper product?

- What is a General Public License (GPL)? Why were MySQL's customers willing to pay for the product, when they could get the product for free under a GPL?

Akamai Case

What is a Content Distribution Network (CDN)? What does it provide over ordinary web caching?

Where did Akamai locate its servers? What barriers to entry existed, for a new entrant to build a CDN to compete with Akamai?

Did Akamai choose to market its products with a direct sales force or through distribution partners? What are the advantages of each choice?

Why did Akamai's marketing strategy have to change when they transitioned from the Free Flow product to the Edge Suite product?
American Airlines Case

How did the dispatch automation package assist the flight dispatchers?

What were some of the benefits of good architecture and Object Oriented Programming in the context of the flight dispatch automation package?