TIM 50 - Business Information Systems

Lecture 14

Instructor: John Musacchio

UC Santa Cruz

Nov 8, 2016
Class Announcements

- Assignment 4
  - Due Now

- Business paper draft due in this week!
  - Due Thursday 11/10

- Database Assignment 2 posted
  - Due Tuesday 11/22
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 lockin

Governments like them because they
- promote competition in some circumstances
- May believe they can be used to national advantage
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?
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: Seatback system to sell seat swaps

- Business Case:
  - Increase revenue, passenger satisfaction
Conceptualization

- New in-flight seatback system
  - Sell upgrades and seat swaps
    - (People who want to get away from sick people ...)
  - Offer to exchange seats
2) Analysis

- Describe what the application will do.
- Enough info to allow “stakeholders” to review idea
- Don’t make highly detailed specifications
- Describe scenarios in which it is used
  - (Use Cases)
2) Analysis -- Example

**Example: Scenario:**

- **Seat Trade**
  - Passenger in 10C (aisle) offers to trade seat for frequent flyer miles
  - Business traveller in 20B (middle) offers to pay $500 to get aisle seat
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.
Architecture

HEADQUARTERS
Airline Dataserver

Wireless Link

Seat back devices
Wireless Link
servers
3) Architecture Continued

- Define the functionality, interaction and interfaces of subsystems
- While doing this, consider
  - **Scalability**
    - How easily can we increase the number of users and maintain performance?
  - **Extensibility**
    - How easily can we add new features in the future?
  - **Administration**
    - How much work will it take by humans to keep this running properly?
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 – offline test of prototype

- Beta test – test in intended environment with cooperative users
  - Example – give HHC to initial group of FA’s
6) Deployment

- Convert from previous processes if necessary
  - Example: CISCO ERP (all at once)
  - Or, you could do incrementally

- Train users

- 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.)
The Database Approach to Data Management

- **Database:**
  - Collection of related files containing records on people, places, or things.
  - Prior to dig. DBs, business used paper files.

- **Entity:**
  - Generalized category representing person, place, thing on which we store info.
  - E.g., SUPPLIER, PART

- **Attributes:**
  - Specific characteristics of each entity:
    - SUPPLIER name, address
    - PART description, unit price, supplier
The Database Approach to Data Management

- **Relational database:**
  - Organize data into tables
  - One table for each entity:
    - E.g., (CUSTOMER, SUPPLIER, PART, SALES)
  - **Fields** (columns) store data representing an attribute.
  - Rows store data for separate **records**.
  - **Key field:** uniquely identifies each record.
  - **Primary key:**
    - One field in each table
    - Cannot be duplicated
    - Provides unique identifier for all information in any row
A relational database organizes data in the form of two-dimensional tables. Illustrated here is a table for the entity SUPPLIER showing how it represents the entity and its attributes. Supplier_Number is the key field.

Figure 5-1
The PART Table

<table>
<thead>
<tr>
<th>Part_Number</th>
<th>Part_Name</th>
<th>Unit_Price</th>
<th>Supplier_Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>137</td>
<td>Door latch</td>
<td>22.00</td>
<td>8259</td>
</tr>
<tr>
<td>145</td>
<td>Side mirror</td>
<td>12.00</td>
<td>8444</td>
</tr>
<tr>
<td>150</td>
<td>Door molding</td>
<td>6.00</td>
<td>8263</td>
</tr>
<tr>
<td>152</td>
<td>Door lock</td>
<td>31.00</td>
<td>8259</td>
</tr>
<tr>
<td>155</td>
<td>Compressor</td>
<td>54.00</td>
<td>8261</td>
</tr>
<tr>
<td>178</td>
<td>Door handle</td>
<td>10.00</td>
<td>8259</td>
</tr>
</tbody>
</table>

Figure 5-2

Primary Key

Foreign Key
The Database Approach to Data Management

• Establishing relationships
  • Entity-relationship diagram
    • Used to clarify table relationships in a relational database
  • Relational database tables may have:
    • One-to-one relationship
    • One-to-many relationship
    • Many-to-many relationship
      • Requires creating a table (join table, Intersection relation) that links the two tables to join information
A Simple Entity-Relationship Diagram

This diagram shows the relationship between the entities SUPPLIER and PART.

Figure 5-3
The Database Approach to Data Management

• Normalization
  • Process of streamlining complex groups of data to:
    • Minimize redundant data elements.
    • Minimize awkward many-to-many relationships.
    • Increase stability and flexibility.

• Referential integrity rules
  • Used by relational databases to ensure that relationships between coupled tables remain consistent.
The Database Approach to Data Management

Sample Order Report

<table>
<thead>
<tr>
<th>Order Number</th>
<th>Part Number</th>
<th>Part Quantity</th>
<th>Part Name</th>
<th>Unit Price</th>
<th>Extended Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>3502</td>
<td>137</td>
<td>10</td>
<td>Door latch</td>
<td>22.00</td>
<td>$220.00</td>
</tr>
<tr>
<td>3502</td>
<td>152</td>
<td>20</td>
<td>Door lock</td>
<td>31.00</td>
<td>620.00</td>
</tr>
<tr>
<td>3502</td>
<td>178</td>
<td>5</td>
<td>Door handle</td>
<td>10.00</td>
<td>50.00</td>
</tr>
</tbody>
</table>

Order Total: $890.00

Figure 5-4
The Final Database Design with Sample Records

Figure 5-5
This diagram shows the relationship between the entities SUPPLIER, ART, LINE_ITEM, and ORDER.
Specific type of software for creating, storing, organizing, and accessing data from a database

Separates the logical and physical views of the data

Logical view: how end users view data

Physical view: how data are actually structured and organized

Examples of DBMS: Microsoft Access, DB2, Oracle Database, Microsoft SQL Server, MySQL
Human Resources Database with Multiple Views

Figure 5-7