Class announcements

- Final Exam:
  - Wednesday 12/10, 12-3pm, Kresge 321

Student Presentations

- Jiabi (Stephy) He
- Jeffrey Hogg

Akamai Case

Freeflow

- Deployed in 1999
- Akamai Infrastructure
  - 13000 servers in 954 networks by 2001
- Customers -
  - Large Commercial Websites
- Revenue model - $2000 per mbps served
  - (For comparison, normal Internet access cost 500 mbps at time)
### 2000 Financials
- $196 Million Loss (before special charge)
- $90 million revenue
- %20 gross margin, after deducting
  - server depreciation
  - payments to network partners
  - Data center space
  - But, most expenses of shouldn’t grow at same rate as number of customers, so margin should improve
- $201.5 million SG&A
  - (selling general and administrative)
  - (largely sales force cost)
  - Again, this might not grow at same rate as the number of customers.
- $40 million R&D

### Competition
- Hosting firms (substitute)
  - Exodus
- Other CDNs
  - Sandpiper, Adero, Mirror Image
- Content Alliances
  - Akamai’s competitors banded together to share networks

### 2001 Market Changes
**Bad**
- Dot-coms bust
- Customers leave
  - “churn rate goes to 22% per quarter"
**Good**
- Hosting firms go bust (exodus)
- Some CDN competitors go bust.
- Competing CDN alliances mired in problems

### EdgeSuite
- Assemble dynamic pages at edges rather than just serve heavy objects
- Value proposition
  - Performance improvement
  - Cost and complexity reduction
  - Scalability
  - Security
- Pricing – higher than old service
- Soon edge suite dominated revenue

### Technology
Dynamic CDN technology: ESI (edge sides includes)
Develop as open standard why?
Akamai not big and credible enough to force a de-facto standard on market
Marketing

- Difference in selling old vs new products:
  - Old product
    - Geared toward speeding up websites
    - Revenues of their clients depended on speed
    - Easier to get sale
  - New Product
    - Simplify company IT function
    - Cost vs. revenue center
    - Harder sell. More data driven...
  - Consequently new product needs more professional sales force
- Channels?
  - Distribution Partners (IBM) credibility
  - Direct Sales Force too

Recent Performance

<table>
<thead>
<tr>
<th>Year</th>
<th>Revenue</th>
<th>Net Sales (loss)</th>
<th>Net Income (loss)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>$2,000</td>
<td>$14,000</td>
<td>($5,000)</td>
</tr>
<tr>
<td>2001</td>
<td>$2,100</td>
<td>$15,000</td>
<td>($4,000)</td>
</tr>
<tr>
<td>2002</td>
<td>$2,200</td>
<td>$16,000</td>
<td>($3,000)</td>
</tr>
<tr>
<td>2003</td>
<td>$2,300</td>
<td>$17,000</td>
<td>($2,000)</td>
</tr>
<tr>
<td>2004</td>
<td>$2,400</td>
<td>$18,000</td>
<td>($1,000)</td>
</tr>
</tbody>
</table>

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
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.

2) Analysis

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

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

- Airline Dataserver
- HEADQUARTERS
- Intranet
- HHC Server
- Wireless Link
- HHC
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
  - Offline 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 HHC
- 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.)