Class announcements

- Database Deadline 11/28

- Reading for Tues (11/28):
  - Messerschmitt Ch 10 (293-321)

- Student Presentations Tuesday 11/28
  - Ken Lee (Business Paper)
  - Rex Pechler (Business Paper)
Student Presentations

- Trevor Behnke (Business Paper)
- Aung Zin (Business Paper)
Akamai Case
Internet Bottlenecks

- **First Mile** (Server Capacity) - 70% of website performance problems according to one study

- **Backbone** - Plentiful, but some shortage within metropolitan areas

- **Peering** - Exchange of traffic between NSPs

- **Last Mile** to home
  - 56 K modems are slow
  - Shared LAN limitations
Solutions

- **Expand Bandwidth**
  - Being done

- **Mirroring web cites**
  - Put exact copy of same web page to multiple servers
  - Tricky to duplicate content

- **Caching**
  - Problem: Stale Content
  - Problem: Hard to count “click throughs”

- **Content Distribution Networks...**
Akamai Freeflow

Local Office or ISP

Akamai Server

INTERNET

NSP 1

NSP 2

Large Company

Web Server

Web Page

Text….
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 charges)
- **$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
Local Office or ISP

Large Company

Web Server

Akamai Server

INTERNET

NSP 1

NSP 2

Movies in Santa Cruz after 8pm?

Web Page

Movie A

Movie C

Construct Page

Web Page

Movie A

Movie A

Movie B

Movie C

Movie C

Movie A

Movie C

Construct Page

Movie A

Movie B

Movie C
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
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

## Consolidated Statements of Operations Data:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Revenues</strong></td>
<td>$210,015</td>
<td>$161,259</td>
<td>$144,976</td>
<td>$163,214</td>
<td>$89,766</td>
</tr>
<tr>
<td><strong>Total cost and operating expenses</strong></td>
<td>161,048</td>
<td>172,370</td>
<td>327,580</td>
<td>2,577,108</td>
<td>989,359</td>
</tr>
<tr>
<td><strong>Net income (loss)</strong></td>
<td>34,364</td>
<td>(29,281  )</td>
<td>(204,437 )</td>
<td>(2,435,512)</td>
<td>(885,785)</td>
</tr>
<tr>
<td><strong>Net income (loss) attributable to common stockholders</strong></td>
<td>34,364</td>
<td>(29,281  )</td>
<td>(204,437 )</td>
<td>(2,435,512)</td>
<td>(885,785)</td>
</tr>
</tbody>
</table>

(In thousands, except share data)
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.
  - Present diagram to FA's
  - HHC customer info updated wirelessly at gate
  - Also has reporting function for misbehaving passengers.

- Business Case:
  - Increase repeat business from high value customers.
1) Conceptualization -- Example:
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: Scenarios:**

**NORMAL FUNCTION**
- When at gate, WiFi AP sends pass. data of next flight to HHC
- HHC displays info on color coded seat map
- If FA clicks on seat she gets more info about passenger

**REPORTING FUNCTION**
- FA wants to report that passenger in 13F is bad.
- FA clicks “report pass.” button followed by 13f
- HHC finds from its data that Joe Schmoe is in 13f
- When HHC is in radio range of WiFi AP, HHC tells server that Joe Schmoe is bad.
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

HHC Server

HEADQUARTERS
Airline Dataserver
Airline Intranet

Wireless Link

HHC
Design a hierarchical architecture.
Again, we see layering and hierarchy.
Between each module we specify an interface.
Our architecture makes use of the existing interface of the airline database, so we don’t need to redesign it!
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?
    - (Remember Sun thin vs fat client discussion)
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
  - 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.)
Markup languages
A **markup language** describes the structure of a document

- Based on tags
- Tags denote structural elements like sections, subsections, figures, etc

Internationally standardized, so application independent
Example: HTML

<html>
<h1>Super Widget</h1>
<h2>Widgets Incorporated</h2>
<em>123456789</em>
<br/>
<p>$300</p>
</html>
Example: XML

Tags Emphasize what the things *mean* rather than how to *format* their Presentation.

```xml
<xml>
  <product>
    <model>Super Widget</model>
    <make>Widgets Incorporated</make>
    <sku>123456789</sku>
    <price>$300</price>
  </product>
</xml>
```
XML in Ecommerce example

Stuff4U

<table>
<thead>
<tr>
<th>Product</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Super Widget</td>
<td>$300</td>
</tr>
<tr>
<td>Amazing Gadget</td>
<td>$500</td>
</tr>
</tbody>
</table>

Supplier

Product info

From each Supplier sent in XML

Retailer

Consumer
XML in ecommerce example 2

Suppliers send product info in XML format. Each Supplier sends the following XML information:

```
<xml>
  <product>
    <model> Super Widget </model>
    <make> Widgets Incorporated </make>
    <sku> 123456789 </sku>
    <price> $300 </price>
  </product>
<xml>
```

The Super widget is recognized and managed by SCM software.

XYZ Manufacturing
Family lineage

- **SGML**
  - Standardized in mid 80s by ISO
  - Introduced in Early 90s
  - Emphasizes formatting and presentation of documents

- **HTML**
  - Introduced in Early 90s
  - Emphasizes structure of documents

- **XML**
  - Proposed in mid 90s
  - Purpose- and industry-specific extensions