IMPORTANT NOTICE TO STUDENTS

These slides are NOT to be used as a replacement for student notes. These slides are sometimes vague and incomplete on purpose to spark class discussions.

Enterprise
Web-based Software
Architecture & Design

CS 446/646 ECE452
Jun 6th, 2011
Characteristics

Servers
- application server
- web server
- proxy servers

Platforms & Technologies
- heterogeneous

Focus/Scope
- usually well defined

Clients
- heterogeneous
  - users, business partners (B2B)
- scale
  - large number of clients
- distributed
Characteristics

Data

- large amounts of data
- long-term & short term persistence
- distributed in nature
- governed by schema
  - global company wide
  - local application specific
  - complex & resistant to change (why?)
Architectural Style

Layered Style

Are you sure?
Architectural Style

Tiered Style

- from layered to tiered
  - physical separation
  - each tier
    - acts as a client of the tier to the right
    - provides a service to the tier on the left

- consequences?
Client-Server Style

- distributed heterogeneous clients
  - thick & thin
  - isolated from each other
- centralized servers
  - computationally powerful
  - one server to support many clients
- design considerations?
Architectural Style

Client-Server Style

- observations
  - main functionality processed at the central server
  - user interface at each client
  - flows
    - data flows from server to client
    - control flows from client to server
- did some body say data?
Architectural Style

Repository Style

- central repository
  - multiple data-sources
  - generally database type (why?)

- data is shared across
  - clients
  - applications

- data is dynamic
Functional Concerns

Application \{functional components\}

- collection of business functionality
- generally divided over \textbf{multiple} tiers

Data

- transactional
  - ACID
    - \textbf{atomicity} – all or nothing
    - \textbf{consistency} – from one consistent state to another consistent state
    - \textbf{isolation} – interaction of other operations with the modified data
    - \textbf{durability} – data after a successful transaction is never lost
Non-functional Concerns

- concurrency
- availability
- security
- performance
- fault-tolerance
- application distribution & deployment
- evolution
- re-usability

Mostly Honoured
Non-functional Concerns

- cost
- ease of use
- interoperability
- portability
- throughput

Mostly commonly violated
Why web applications?

- what non-functional requirements are we solving?
Web-based Enterprise Apps

Why web applications?

- what non-functional requirements are we solving?

| concurrency | evolution |
| availability | re-usability |
| security | cost |
| performance | ease of use |
| fault-tolerance | interoperability |
| application distribution | portability |
| application deployment | throughput |
Web-based Enterprise Apps

Key Attributes

• thin clients – web browsers
  – computationally challenged

• user interface – HTML, javascript, css
  – simple & static (mostly)
  – resides at client tier

• communication
  – synchronous request response cycle
  – HTTP over TCP/IP
  – what about data
First Generation

HTTP request
(URL or Form posting)

Client Tier

Server Tier

Data Tier

HTTP response
(HTML Document)

CGI-Scripts

Data Source

Web

WATERLOO
CHERITON SCHOOL OF
COMPUTER SCIENCE
First Generation

Observations

• simple design
• client-tier
  – building blocks are?
• business tier
  – aggregation of scripts
  – scripts are
    • independent
    • stateless
• lacks organic growth (how?)
• security nightmare
Second Generation

HTTP request
(URL or Form posting)

HTTP response
(HTML Document)

Client Tier

Server Tier

Data Tier

Presentation

JEE Container

Servlet
Services
JDBC

Servlet

Data Source

WATERLOO
CHERITON SCHOOL OF
COMPUTER SCIENCE
Second Generation

Observations

- not so simple anymore
- **improves business tier** only
  - high level frameworks
    - JEE servlets, struts, spring MVC
  - applications server standardization
    - design by contract
    - provides various services (*like what?*)
- negative impact on
  - **request-response cycle** (*why?*)
  - user interface (*why?*)