INTERNET & WEB APPLICATION DEVELOPMENT

SWE 444

Fall Semester 2008-2009 (081)

Module 6: Web Engineering Fundamentals

Dr. El-Sayed El-Alfy
Computer Science Department
King Fahd University of Petroleum and Minerals
alfy@kfupm.edu.sa

Objectives/Outline

- Objectives
  - Understand the role of web engineering
  - Learn a systematic process for web applications development

- Outline
  - Introduction
  - Requirements Analysis
  - Web Modeling
  - Web Design and Architecture
  - Web Accessibility

Resources

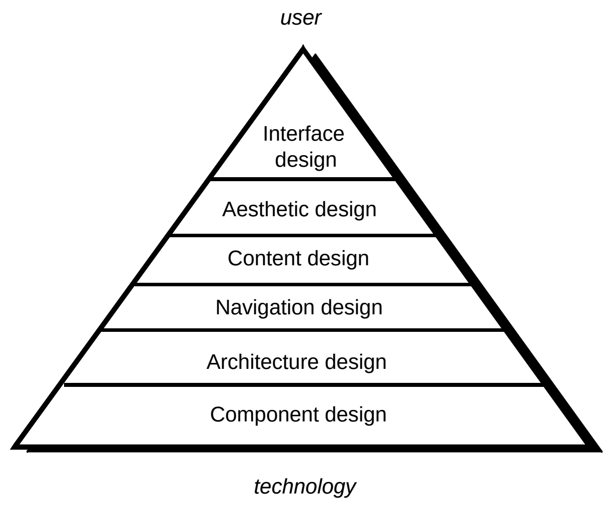
- Books
  - Roger S. Pressman, David Lowe (2009). *Web Engineering: A Practitioner's Approach*, McGraw-Hill. http://highered.mcgraw-hill.com/sites/0073523291/
  - Roger Pressman (2005). *Software Engineering: A Practitioner's Approach*, 6/e, McGraw-Hill Higher Education. Chapters 16-20. http://highered.mcgraw-hill.com/sites/0072853182/information_center_view0/
  - G. Kappel, B. Pröll, S. Reich, and W. Retschitzegger (eds), *Web Engineering - The Discipline of Systematic Development of Web Applications*, John Wiley & Sons, 2006. http://www.web-engineering.at/eng/
- Online material
  - INFSCI 2955: Web Engineering
  - Department of Information Science and Telecommunications, University of Pittsburgh http://www.sis.pitt.edu/~jgrady/

6.4 WEB APPLICATION DESIGN

Outline

- Design Pyramid
- Design Goals
- Design Metrics
- Interface Design
- Aesthetic Design
- Content Design
- Navigation Design
- Architecture defined
- Developing architectures
- Types of architectures
- Generic Web Architecture
- Layered-aspect architectures
- Data-aspect architectures

Design Pyramid



Design Goals

- Consistency
  - Content should be constructed consistently
  - Graphic design (aesthetics) should present a consistent look across all parts of the WebApp
  - Architectural design should establish templates that lead to a consistent hypermedia structure
  - Interface design should define consistent modes of interaction, navigation and content display
  - Navigation mechanisms should be used consistently across all WebApp elements

Design Goals (cont.)

- Identity
  - Establish an "identity" that is appropriate for the business purpose
- Robustness
  - The user expects robust content and functions that are relevant to the user's needs
- Navigability
  - designed in a manner that is intuitive and predictable
- Visual appeal
  - the look and feel of content, interface layout, color coordination, the balance of text, graphics and other media, navigation mechanisms must appeal to end-users
- Compatibility
  - With all appropriate environments and configurations

Design Goals (cont.)

- Security
  - Rebuff external attacks
  - Exclude unauthorized access
  - Ensure the privacy of users/customers
- Availability
  - the measure of the percentage of time that a WebApp is available for use
- Scalability
  - Can the WebApp and the systems with which it is interfaced handle significant variation in user or transaction volume
- Time to Market

Design Metrics

- Does the user interface promote usability?
- Are the aesthetics of the WebApp appropriate for the application domain and pleasing to the user?
- Is the content designed in a manner that imparts the most information with the least effort?
- Is navigation efficient and straightforward?
- Has the WebApp architecture been designed to accommodate the special goals and objectives of WebApp users, the structure of content and functionality, and the flow of navigation required to use the system effectively?
- Are components designed in a manner that reduces procedural complexity and enhances the correctness, reliability and performance?

Interface Design

- Where am I? The interface should
  - provide an indication of the WebApp that has been accessed
- inform the user of her location in the content hierarchy.
- What can I do now? The interface should always help the user understand his current options
  - what functions are available?
  - what links are live?
  - what content is relevant?
- Where have I been, where am I going? The interface must facilitate navigation.
  - Provide a "map" (implemented in a way that is easy to understand) of where the user has been and what paths may be taken to move elsewhere within the WebApp.

Effective Interfaces

- Bruce Tognozzi [TOG01] suggests…
  - Effective interfaces are visually apparent and forgiving, instilling in their users a sense of control. Users quickly see the breadth of their options, grasp how to achieve their goals, and do their work.
  - Effective interfaces do not concern the user with the inner workings of the system. Work is carefully and continuously saved, with full option for the user to undo any activity at any time.
  - Effective applications and services perform a maximum of work, while requiring a minimum of information from users.

Interface Design Principles

- Anticipation—A WebApp should be designed so that it anticipates the use's next move.
- Communication—The interface should communicate the status of any activity initiated by the user
- Consistency—The use of navigation controls, menus, icons, and aesthetics (e.g., color, shape, layout)
- Controlled autonomy—The interface should facilitate user movement throughout the WebApp, but it should do so in a manner that enforces navigation conventions that have been established for the application.
- Efficiency—The design of the WebApp and its interface should optimize the user's work efficiency, not the efficiency of the Web engineer who designs and builds it or the client-server environment that executes it.

Interface Design Principles (cont.)

- Focus—The WebApp interface (and the content it presents) should stay focused on the user task(s) at hand.
- Fitt's Law—"The time to acquire a target is a function of the distance to and size of the target."
- Human interface objects—A vast library of reusable human interface objects has been developed for WebApps.
- Latency reduction—The WebApp should use multi-tasking in a way that lets the user proceed with work as if the operation has been completed.
- Learnability— A WebApp interface should be designed to minimize learning time, and once learned, to minimize relearning required when the WebApp is revisited.

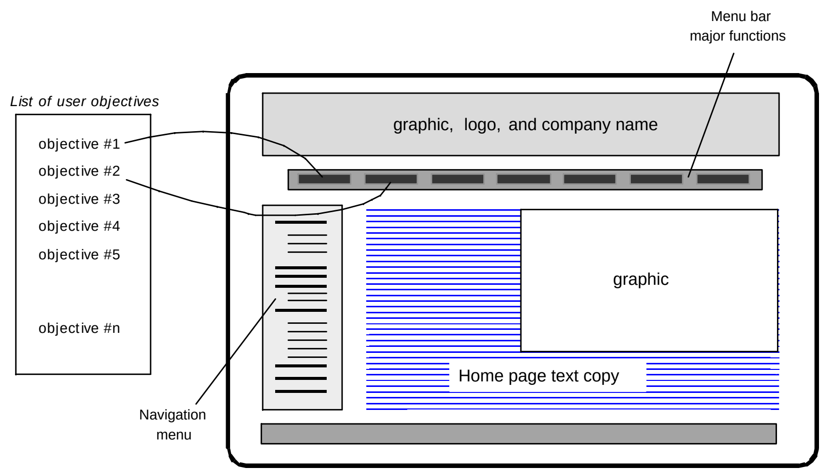
Interface Design Principles (cont.)

- Maintain work product integrity—A work product (e.g., a form completed by the user, a user specified list) must be automatically saved so that it will not be lost if an error occurs.
- Readability—All information presented through the interface should be readable by young and old.
- Track state—When appropriate, the state of the user interaction should be tracked and stored so that a user can logoff and return later to pick up where she left off.
- Visible navigation—A well-designed WebApp interface provides "the illusion that users are in the same place, with the work brought to them."

Interface Design Workflow

- Review information contained in the analysis model and refine as required.
- Develop a rough sketch of the WebApp interface layout.
- Map user objectives into specific interface actions.
- Define a set of user tasks that are associated with each action.
- Storyboard screen images for each interface action.
- Refine interface layout and storyboards using input from aesthetic design.
- Identify user interface objects that are required to implement the interface.
- Develop a procedural representation of the user's interaction with the interface.
- Develop a behavioral representation of the interface.
- Describe the interface layout for each state.
- Refine and review the interface design model.

Mapping User Objectives



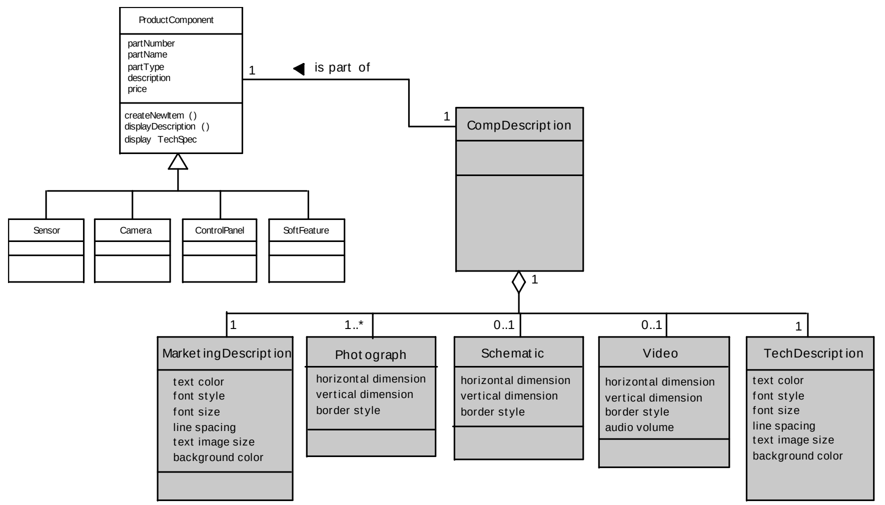
Aesthetic Design

- Don't be afraid of white space.
- Emphasize content.
- Organize layout elements from top-left to bottom right.
- Group navigation, content, and function geographically within the page.
- Don't extend your real estate with the scrolling bar.
- Consider resolution and browser window size when designing layout.

Content Design

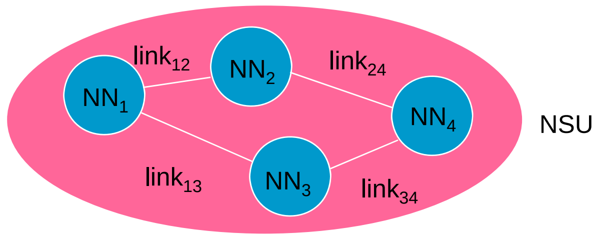
- Develops a design representation for content objects
  - For WebApps, a content object is more closely aligned with a data object for conventional software
- Represents the mechanisms required to instantiate their relationships to one another.
  - analogous to the relationship between analysis classes and design components
- A content object has attributes that include content-specific information and implementation-specific attributes that are specified as part of design

Design of Content Objects

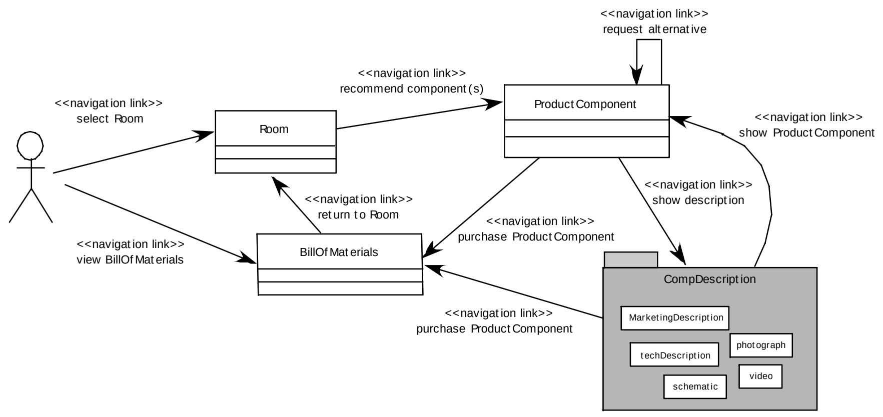


Navigation Design

- Begins with a consideration of the user hierarchy and related use-cases
  - Each actor may use the WebApp somewhat differently and therefore have different navigation requirements
- As each user interacts with the WebApp, she/he encounters a series of navigation semantic units (NSUs)
  - NSU—"a set of information and related navigation structures that collaborate in the fulfillment of a subset of related user requirements"
  - Ways of navigation (WoN)—represents the best navigation way or path for users with certain profiles to achieve their desired goal or sub-goal. Composed of …
    - Navigation nodes (NN) connected by Navigation links



Creating an NSU: Example


<<navigation link>> request alternative
<<navigation link>> recommend component(s)
Product Component
<<navigation link>> select Room
Room
<<navigation link>> show Product Component
<<navigation link>> return to Room
<<navigation link>> show description
<<navigation link>> purchase Product Component
<<navigation link>> view BillOfMaterials
BillOfMaterials
CompDescription
<<navigation link>> purchase Product Component
MarketingDescription
photograph
techDescription
video
schematic

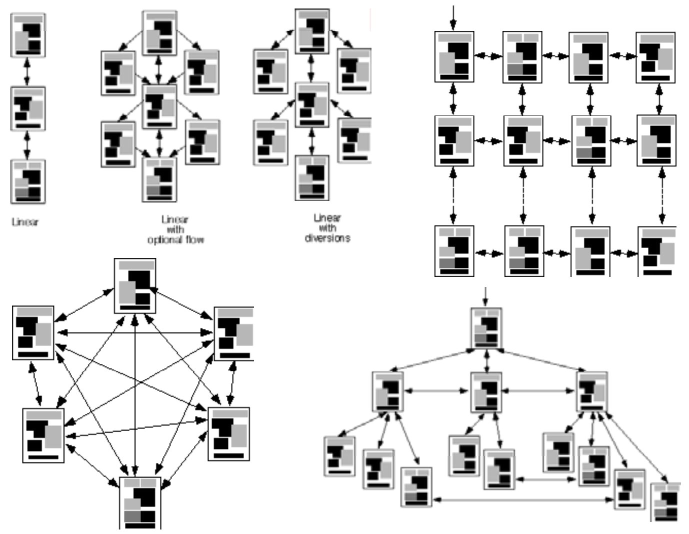
Navigation Syntax

- Individual navigation link—text-based links, icons, buttons and switches, and graphical metaphors..
- Horizontal navigation bar—lists major content or functional categories in a bar containing appropriate links. In general, between 4 and 7 categories are listed.
- Vertical navigation column
  - lists major content or functional categories
  - lists virtually all major content objects within the WebApp.
- Tabs—a metaphor that is nothing more than a variation of the navigation bar or column, representing content or functional categories as tab sheets that are selected when a link is required.
- Site maps—provide an all-inclusive tab of contents for navigation to all content objects and functionality contained within the WebApp.

Architecture Design

- Define "software architecture"
  - http://www.sei.cmu.edu/architecture/definitions.html
  - *"Software architecture is the set of design decisions which, if made incorrectly, may cause your project to be cancelled."* – Eoin Woods
- Authors focus on 5 key attributes of software architectures
  - Structure, Elements, Relationships
  - Analysis => Implementation
  - Multiple viewpoints (conceptual, runtime, process & implementation)
  - Understandable
  - Framework for flexibility
- Content architecture focuses on the manner in which content objects (or composite objects such as Web pages) are structured for presentation and navigation.
  - The term information architecture is also used to connote structures that lead to better organization, labeling, navigation, and searching of content objects.
- WebApp architecture addresses the manner in which the application is structured to manage user interaction, handle internal processing tasks, effect navigation, and present content.
- Architecture design is conducted in parallel with interface design, aesthetic design and content design.

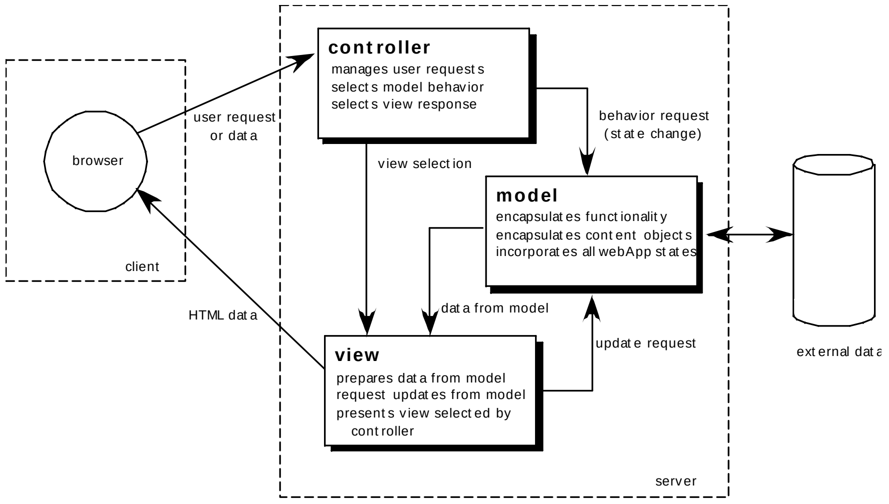
Content Architecture



MVC Architecture

- M: The model contains all application specific content and processing logic, including
  - all content objects
  - access to external data/information sources,
  - all processing functionality that are application specific
- V: The view contains all interface specific functions and enables
  - the presentation of content and processing logic
  - access to external data/information sources,
  - all processing functionality required by the end-user.
- C: The controller manages access to the model and the view and coordinates the flow of data between them.

MVC Architecture (cont.)

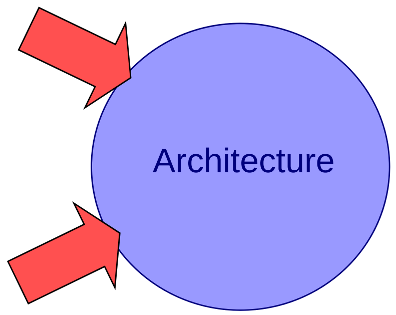


Developing Architectures

- Influences on Architectures

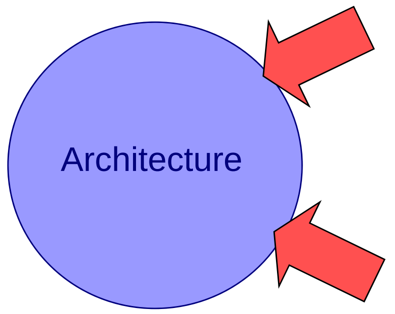
__Functional Requirements__
•Clients
•Users
•Other Stakeholders

__Experience with__
•Existing Architecture
•Patterns
•Project Management
•Other?



Developing Architectures

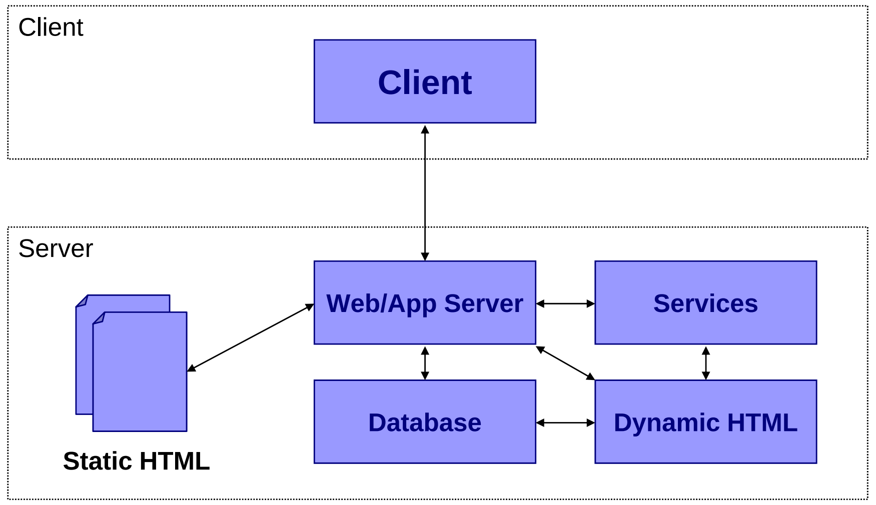
- Influences on Architectures (continued)



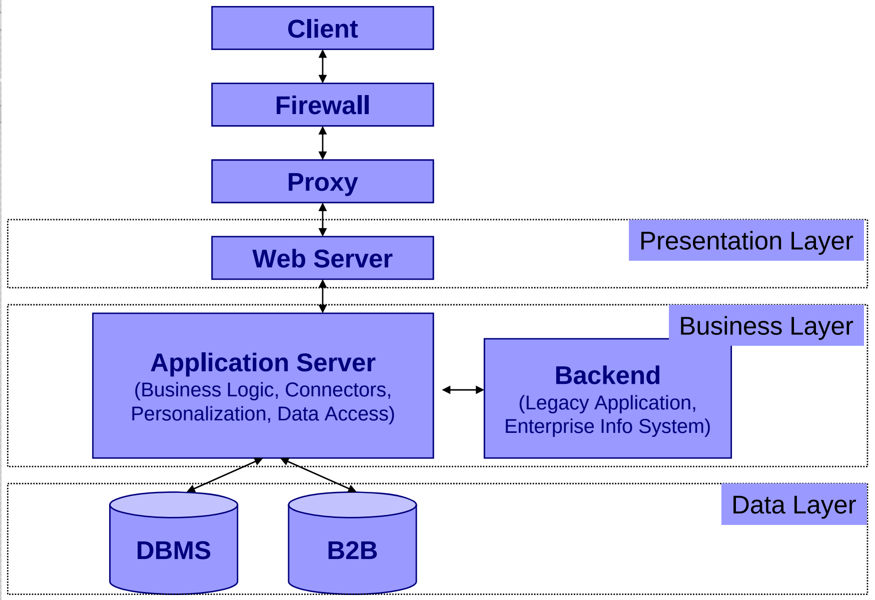
Quality considerations with
- •Performance
- •Scalability
- •Reusability
- •Other?

Technical Aspects
- •Operating System
- •Middleware
- •Legacy Systems
- •Other?

Client/Server (2-Layer)



N-Layer Architectures



Why an N-Layer Architecture?

- Separating services in business layer promotes re-use different applications
  - Loose-coupling – changes reduce impact on overall system.
  - More maintainable (in terms of code)
  - More extensible (modular)
- Trade-offs
  - Needless complexity
  - More points of failure

More on Proxies

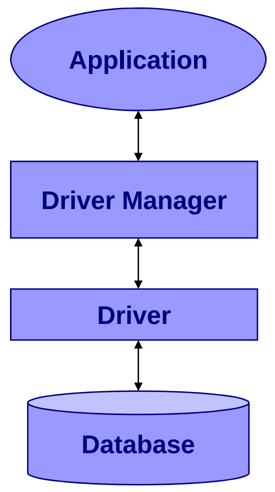
- Originally for *caching* data
- Can also serve other roles:
  - Link Proxy
    - *Persistent URL's* – maps the URL the client sees to the actual URL.
    - *AJAX* – allows data from a 2nd server to be accessed via a client script.
  - History Proxy
    - HTTP is stateless - navigation history cannot be shared across multiple websites.
    - Multiple companies can access a server-side cookie (e.g. DoubleClick)

Integration Architectures

- Enterprise Application Integration (EAI)
- Web Services
- Portals/Portlets
- Challenges/Pitfalls
  - Cannot separate logic & data in legacy systems
  - Incompatible schemes
  - Poor documentation
  - Measuring performance/scalability

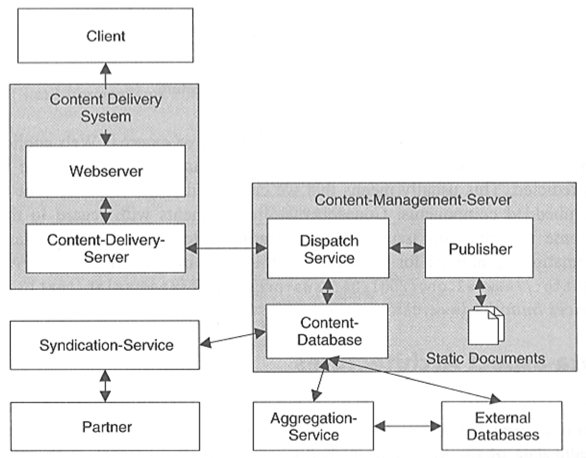
Data-Aspect Architectures

- Data can be grouped into either of 3 architectural categories:
  1. Structured data of the kind stored in DBs
  2. Documents of the kind stored in document management systems
  3. Multimedia data of the kind stored in media servers
- Structured data (JDBC/ODBC)
  - Accessed either directly via a web extension (for 2-tier) or over app server (for n-tier).
  - Since DB technology are highly mature, they are easy to integrate
  - Easy to implement
    - APIs are available to access DBs (e.g., JDBC, ODBC)



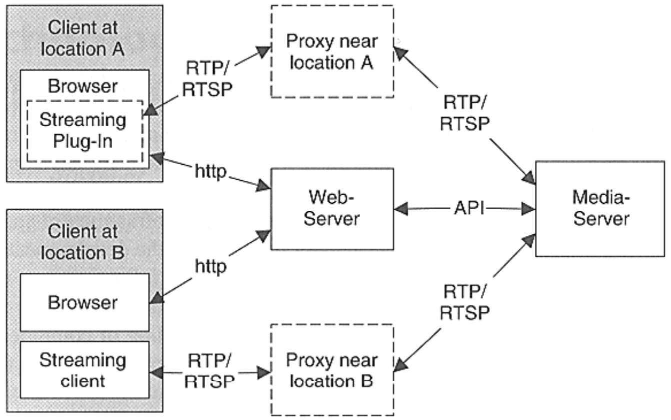
Data-Aspect Architectures

- Web Document Management



Data-Aspect Architectures

- Web Multimedia Management: Point-to-point



Q & A

?