CS603 – Web Engineering

PREPARED BY: DR. REEMA PATEL

Web Engineering

- Web engineering (WebE) as the process used to create high quality Web-based applications (WebApps).
  - WebApps become increasingly integrated in business strategies (e.g. e-commerce)
  - need to build reliable, usable, and adaptable systems grows in importance.

- Web engineering is not a perfect clone of software engineering, but it draws heavily on many of software engineering's principles and management activities.

- The Web engineering process begins with the formulation of the problem to be solved by the WebApp.
  - WebApp requirements are analyzed.
  - Architectural, navigational, and interface design are conducted.
  - The system is implemented using specialized languages and tools associated with the Web.

Web Engineering

- WebApps tend to be highly evolutionary, so mechanisms for configuration management, quality control, and maintenance must be established early.
- Web engineering relies on formal technical reviews to assess the quality of the analysis and design models.
- Specialized reviews are conducted to assess the usability of the WebApp.
- Testing is applied to uncover errors in content, functionality, and compatibility.

Web Application Development

- Challenges and role of web engineering
- The World Wide Web, is increasingly pervading every aspect of our lives.
- Web has dramatically influenced us in several ways and has matured to become a very attractive and dominant platform for deploying business and social applications and organizational information systems.
- It has also become a universal user interface to business applications, information systems, databases, and legacy systems.
- It supports document and workflow management, cooperative work, and distributed knowledge and media (photo, audio, and video) sharing.

Web Application Development

- The growth of the Web has been exponential.
- The interaction between a Web system and its back-end information systems, as well as with other Web systems, has become tighter and complex.
- Many organizations have extended, and still continue to extend, the scope and functionalities of their Web-based applications and are also beginning to provide mobile and wireless access to them.
- Web-based systems and applications now offer an array of content and functionality to a huge population of users and serve many different purposes.
- The Web has become a mainstay, and it is perhaps appropriate to say that our civilization “runs” on the Web as individuals, organizations, and nations rely on a multitude of Web-based systems.

Evolution of the Web and Web Applications

- The Web has evolved beyond anyone's imagination within a short span of years,
  - since Tim Berners-Lee conceived and publicized, on August 6, 1991, a system for turning the Internet into a publishing medium for sharing and dissemination of scientific data and information, which he called the "World Wide Web."
- It has become indispensable and essential to many people and organizations around the world.
- The evolution of the Web has brought together some disparate disciplines such as
  - media, information science, and information and communication technology,
  - facilitating the easy creation, maintenance, sharing, and use of different types of information from anywhere, any time, and
  - using a variety of devices such as desktop and notebook computers, pocket PCs, personal digital assistants (PDAs), and mobile phones.

Classification of Web

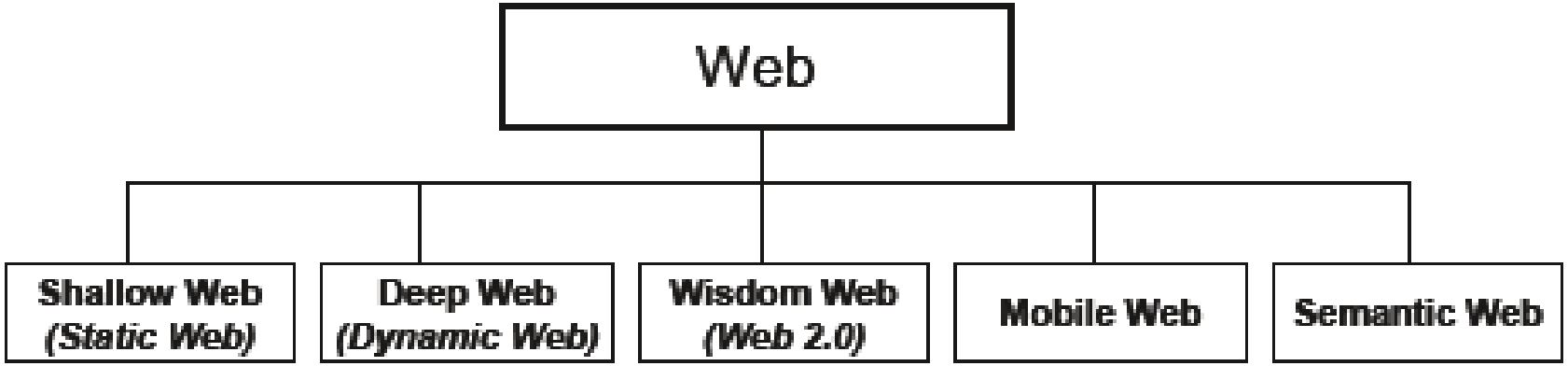


Figure 2.1. Classification of the Web.

Sematic Web

- Berners-Lee originally expressed his vision of the Semantic Web in 1999 as follows:

- I have a dream for the Web [in which computers] become capable of analyzing all the data on the Web – the content, links, and transactions between people and computers. A "Semantic Web", which makes this possible, has yet to emerge, but when it does, the day-to-day mechanisms of trade, bureaucracy and our daily lives will be handled by machines talking to machines. The "intelligent agents" people have touted for ages will finally materialize.

Rich Internet Applications

- Rich Internet applications (RIA) are Web-based applications that run in a Web browser and do not require software installation, but still have the features and functionality of traditional desktop applications.
- The term “RIA” was introduced in a Macromedia whitepaper in March 2002.
- RIA represents the evolution of the browser from a static request-response interface to a dynamic, asynchronous interface.
- Broadband proliferation, consumer demand, and enabling technologies, including the Web 2.0, are driving the proliferation of RIAs.
- RIAs promise a richer user experience and benefits—interactivity and usability that are lacking in many current applications.

Unique Aspects of Web Applications

- Web applications have certain unique intrinsic characteristics that make Web development different and perhaps more challenging compared to traditional software development.

- In most Web applications, frequency and degree of change are much higher than in traditional software applications,
- in many applications it is not possible to specify fully their entire requirements at the beginning.
- Thus, successfully managing the evolution, change, and newer requirements of Web applications is a major technical, organizational, and management challenge—more demanding than traditional software development.

Unique Aspects of Web Applications

- Web applications are meant to be used by a vast, diverse, remote community of users who have different requirements, expectations, and skill sets.

- Therefore, the user interface and usability features have to meet the needs of a diverse, anonymous user community.

- the number of users accessing a Web application at any time is unpredictable and could vary quite considerable – create performance problems

Unique Aspects of Web Applications

- Web-based applications demand presentation of a variety of content – text, graphics, images, audio, video, and content may also integrated with procedural processing.
- Hence, their development includes the creation and management of the content and their presentation in an attractive manner
- Web-based systems, in general, demand good aesthetic appeal—"look and feel" – easy navigation.
- Web applications, especially those meant for a global audience, need to adhere to many different social and cultural sentiments and national standards—including multiple languages and different systems of units.
- Security and privacy needs of Web-based systems are in general more demanding than those of traditional software.

Unique Aspects of Web Applications

- Web applications need to cope with a variety of display devices and formats and support hardware, software, and networks with vastly varying access speeds.

- Ramifications of failure or dissatisfaction of users of Web-based applications can be much worse than conventional IT systems. Also, Web applications could fail for many different reasons.

Challenges

- Design and development of Web applications for mobile and device independent operations is very complex and challenging,
- Testing and validation of Web applications for access by mobile devices is a challenge, as there are many types of devices with varying shapes and sizes.
- the challenge is to design and develop sustainable Web systems for better
  - Comprehension
  - performance (responsiveness)
  - security and integrity
  - evolution, growth, and maintainability
  - Testability
  - mobility

Web System Complexity

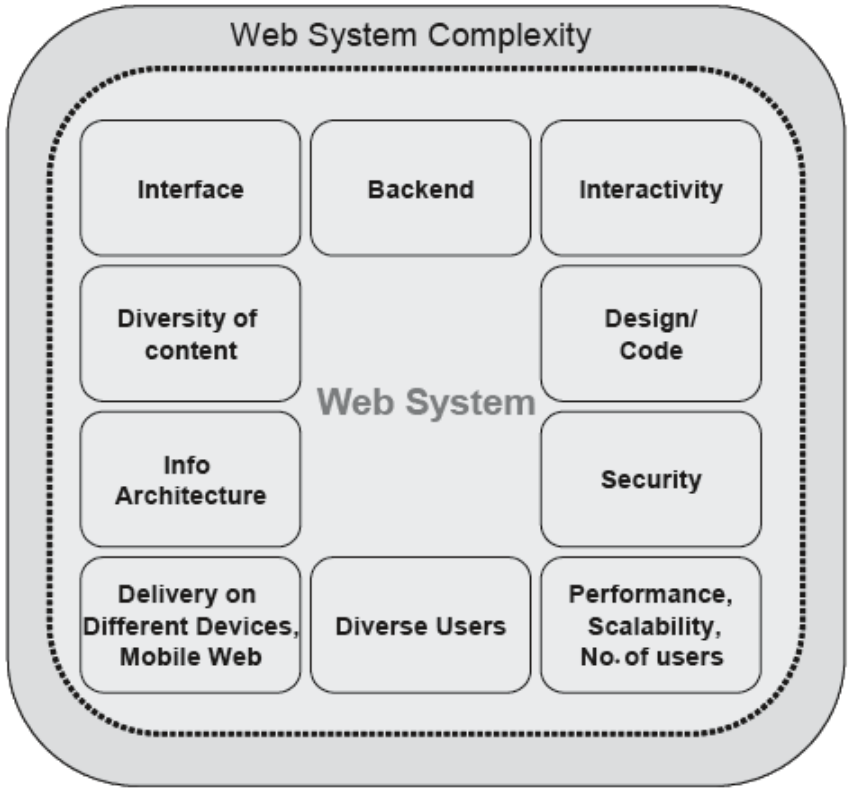


Figure 2.2. Landscape of Web systems.

Web Failures

- Web failures could arise due to many different reasons.
- not meeting functionality and the users' needs
- poor usability
- poor performance
- security breaches
- not functioning properly, including errors and crashes
- poor maintainability
- poor scalability
- schedule and cost over-runs
- abandoned projects—poor project management

Web Engineering

- Web engineering proposes an agile, yet disciplined framework for building industry-quality WebApps.

- Agile:
- Web engineers must understand that modern business demands adaptation, business strategies and rules change rapidly, management demands near-instantaneous responsiveness (even when such demands are completely unreasonable), and stakeholders keep changing their minds even as they demand rapid delivery.
- Customers care about a WebApp that's delivered when they need it, not about the work that goes into creating a deliverable WebApp.
- WebE team must emphasize *agility.*

Web Engineering

- Ivar Jacobson [Jac02] provides a useful discussion of the concept:
- “An agile team is a nimble team able to appropriately respond to changes.
- Change is what software development is very much about.
- Changes in the software being built, changes to the team members, changes because of new technology, changes of all kinds that may have an impact on the product they build or the project that creates the product.
- Support for changes should be built-in everything we do in software, something we embrace because it is the heart and soul of software.
- An agile team recognizes that software is developed by individuals working in teams and that the skills of these people, their ability to collaborate is at the core for the success of the project.”

Web Engineering

- WEbE framework
- A framework establishes the foundation for a complete Web engineering process by identifying a small number of framework activities that are applicable to all WebApp projects, regardless of their size or complexity.

Web Engineering

- Architecture presents a framework, describes the structure, and makes the system understandable. It helps in making the transition from analysis to implementation (Eichinger, 2006).
- In Web system architecture design, various components of the system and how they are linked are decided.
- Web subsystem architecture is composed of the following:
  - an overall system architecture that describes how the network and the various servers such as Web servers, application servers, and database servers interact
  - an application architecture that depicts various information modules and the functions available
  - a software architecture that identifies various software and database modules required to implement the application architecture

Web Engineering

- Web Testing and Evaluation:

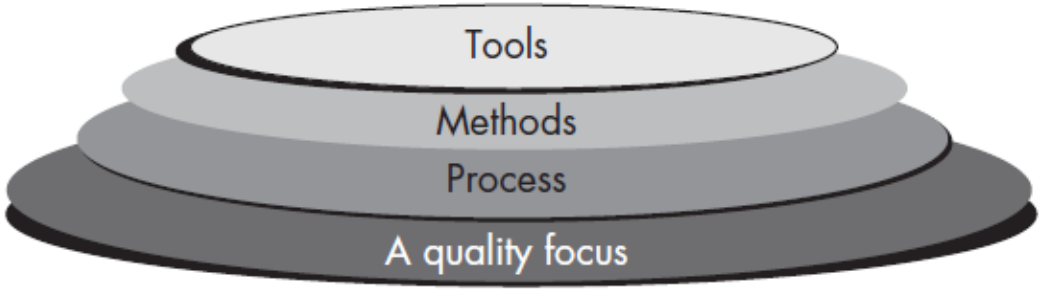
1. browser compatibility
2. page display
3. session management
4. usability
5. content analysis
6. availability
7. backup and recovery
8. transactions
9. shopping, order processing
10. internalization
11. operational business procedures
12. system integration
13. performance
14. login and security

Components of Web Engineering

- Software Engineering
- In a virtual round table published in IEEE Software [Pre98], Roger staked out a position that couples Web engineering and software engineering:
- “It seems to me that just about any important product or system is worth engineering. Before you start building it, you’d better understand the problem, design a workable solution, implement it in a solid way, and test it thoroughly. You should probably also control changes to it as you work and have some mechanism for ensuring the end result’s quality. Many Web developers don’t argue with this; they just think their world is really different and that conventional software engineering approaches simply don’t apply.”

Components of Web Engineering

- Software Engineering Layers:



Web Engineering

- A well-engineered Web system (Murugesan and Ginige, 2005)
  - is functionally complete and correct
  - is usable
  - is robust and reliable
  - is maintainable
  - is secure
  - performs reasonably even under flash and peak loads
  - is scalable
  - is portable, where required (perform across different commonplatforms), compatible with multiple browsers
  - is reusable
  - is interoperable with other systems disabilities
  - is well-documented
  - has universal accessibility

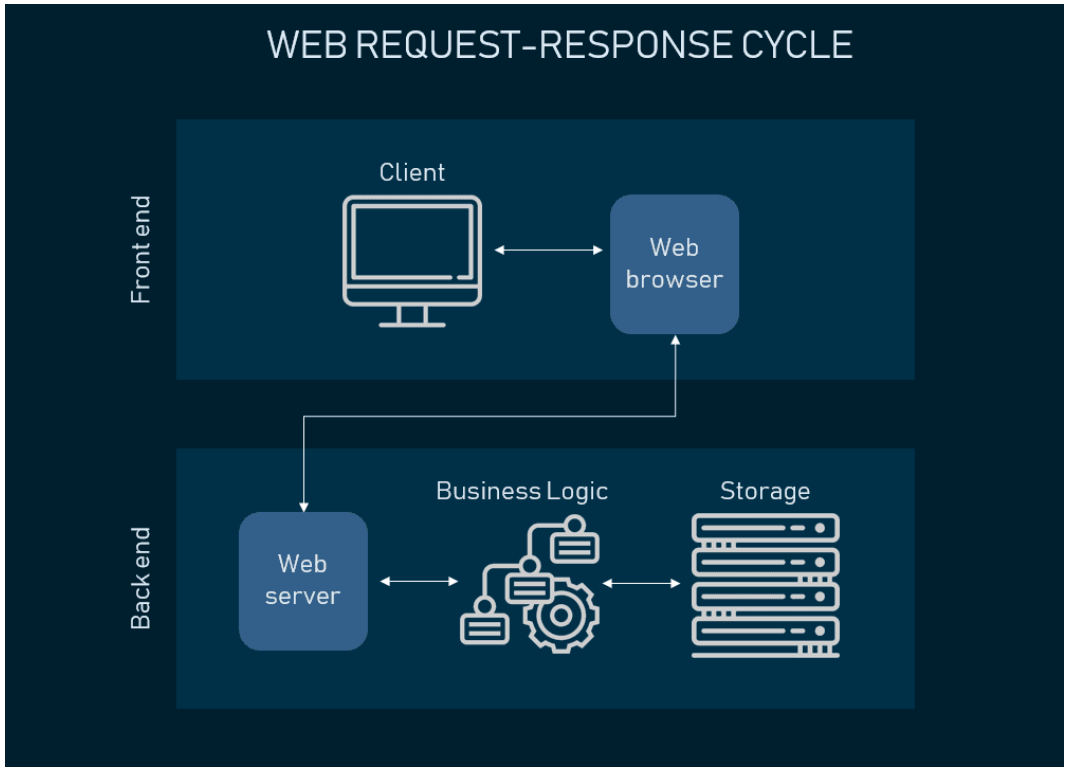
Web Application Architectures

- The quality of a Web application is considerably influenced by its underlying architecture.
- Poor performance, insufficient maintainability and expandability, and low availability of a Web application are often caused by an inappropriate architecture.
- technical constraints like available Web servers, application servers used, or the integration of legacy systems, the architectures of Web applications should also consider the organizational framework in which they are embedded,
  - e.g., the architect's experience.
- The use of flexible multi-layer architectures, the consideration of multimedia contents, and the integration of existing data repositories and applications are challenges in the development of successful architectures for Web applications.

What is an Architecture?

- There is no unique definition of the term “architecture”.
- **Web application architecture** is a mechanism that determines how application components communicate with each other.
  - way the client and the server are connected is established by web application architecture.

How does the web request work?



What is an Architecture?

- **Most important properties of software architectures:**
- ***Architecture describes structure*:**
  - the decomposition into components, and their interfaces and relationships.
  - It describes both the static and the dynamic aspects of that software system,
  - so that it can be considered a building design and flow chart for a software product.

- **Architecture forms the transition from analysis to implementation:**
  - Unified Process: break the functional requirements and quality requirements down into software components and their relationships and interfaces in an iterative approach.

What is an Architecture?

- **Architecture can be looked at from different viewpoints** (supported by UML)**:**
  - conceptual view - identifies entities of the application domain and their relationships
  - runtime view - describes the components at system runtime, e.g., servers, or communication connections;
  - process view - maps processes at system runtime, while looking at aspects like synchronization and concurrency
  - implementation view - describes the system's software artifacts, e.g., subsystems, components, or source code.

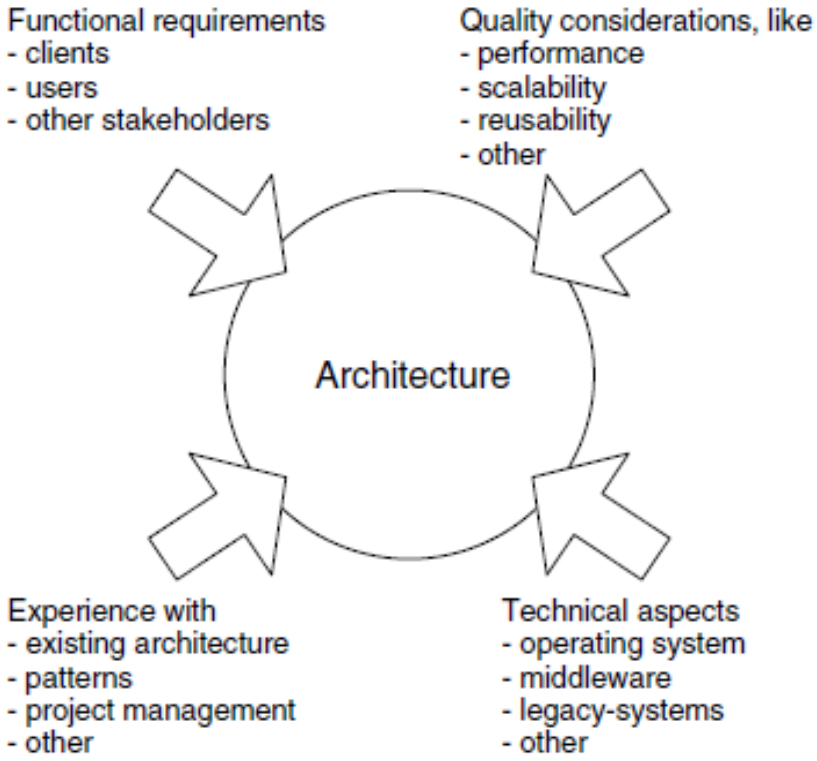
What is an Architecture?

- **Architecture makes a system understandable:**
  - Structuring software systems and breaking them down into different perspectives allows us to better manage the complexity of software systems
  - the abstraction of system aspects facilitates the communication of important architectural issues.

- **Architecture represents the framework for a flexible system:**
  - the software architecture forms the framework in which a software system can evolve.
  - If extensions of a system have not been accounted for in advance, then such an extension will at best be difficult to realize

Developing Architectures

- Figure shows the different factors and constraints influencing the development of an architecture



Patterns

- patterns enable us to reuse proven and consolidated design knowledge, supporting the development of high-quality software systems.
- patterns on three different abstraction levels:
- **Architecture patterns:**
  - They describe architectural subsystems, their responsibilities, relationships, and interplay.
  - Example: Model-View-Controller (MVC) pattern
- **Design patterns:**
  - These patterns describe the structure, the relationships, and the interplay between components to solve a design problem within a defined context.
  - Design patterns abstract from a specific programming language,
  - Example: Publisher-Subscriber pattern

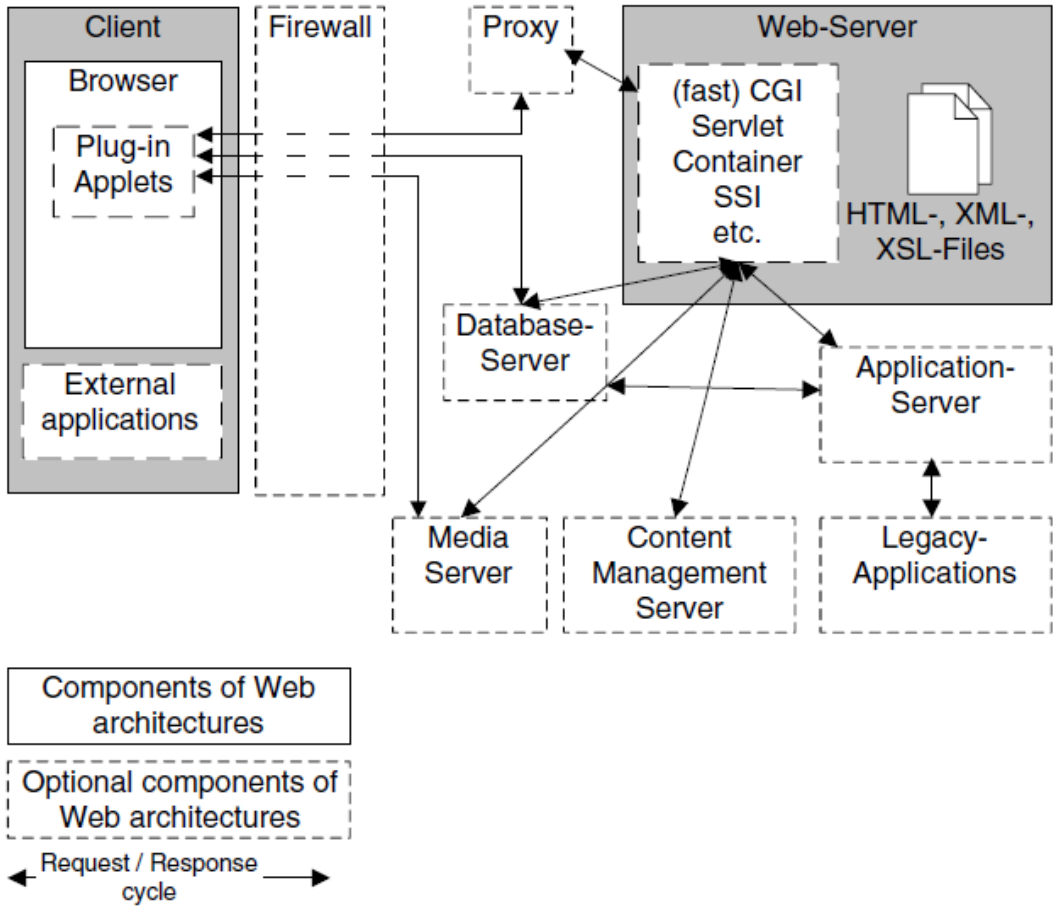
Patterns

- **Idioms:**
  - describe patterns that refer to a specific implementation in a programming language,
  - Example - Counted-Pointer idiom for storage management in C++

Frameworks

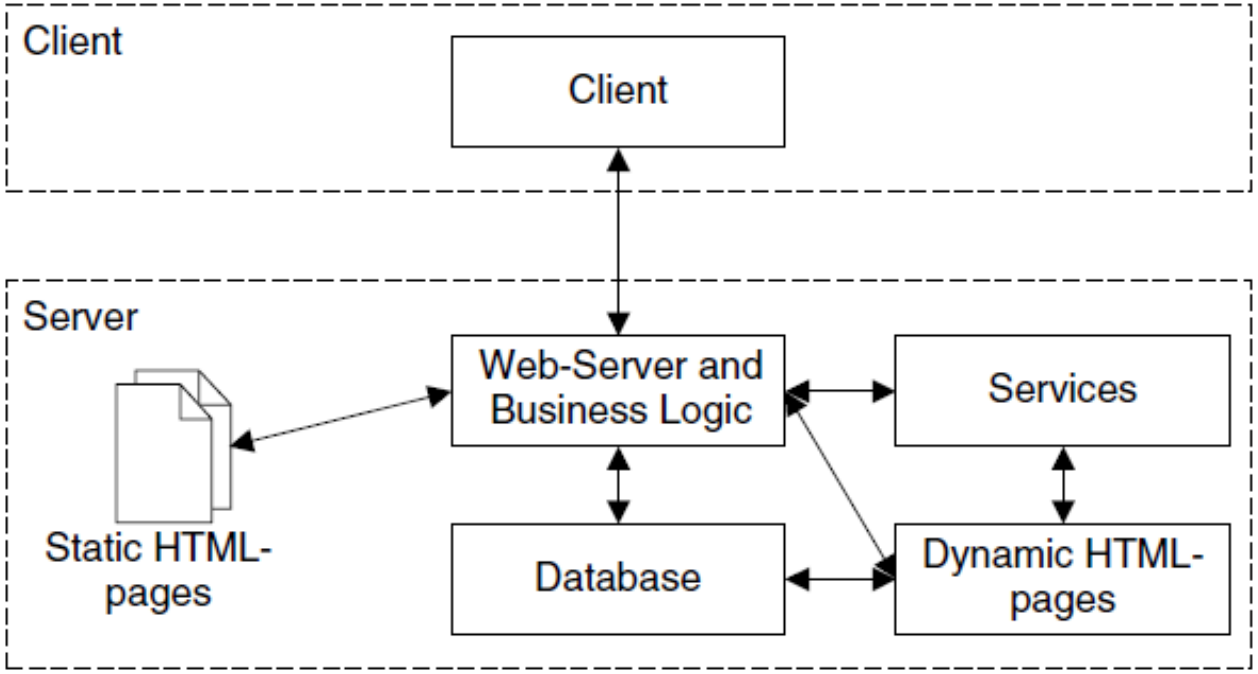
- framework is a reusable software system with general functionality already implemented.
- The framework can be specialized into a ready-to-use application

Components of a Generic Web Application Architecture



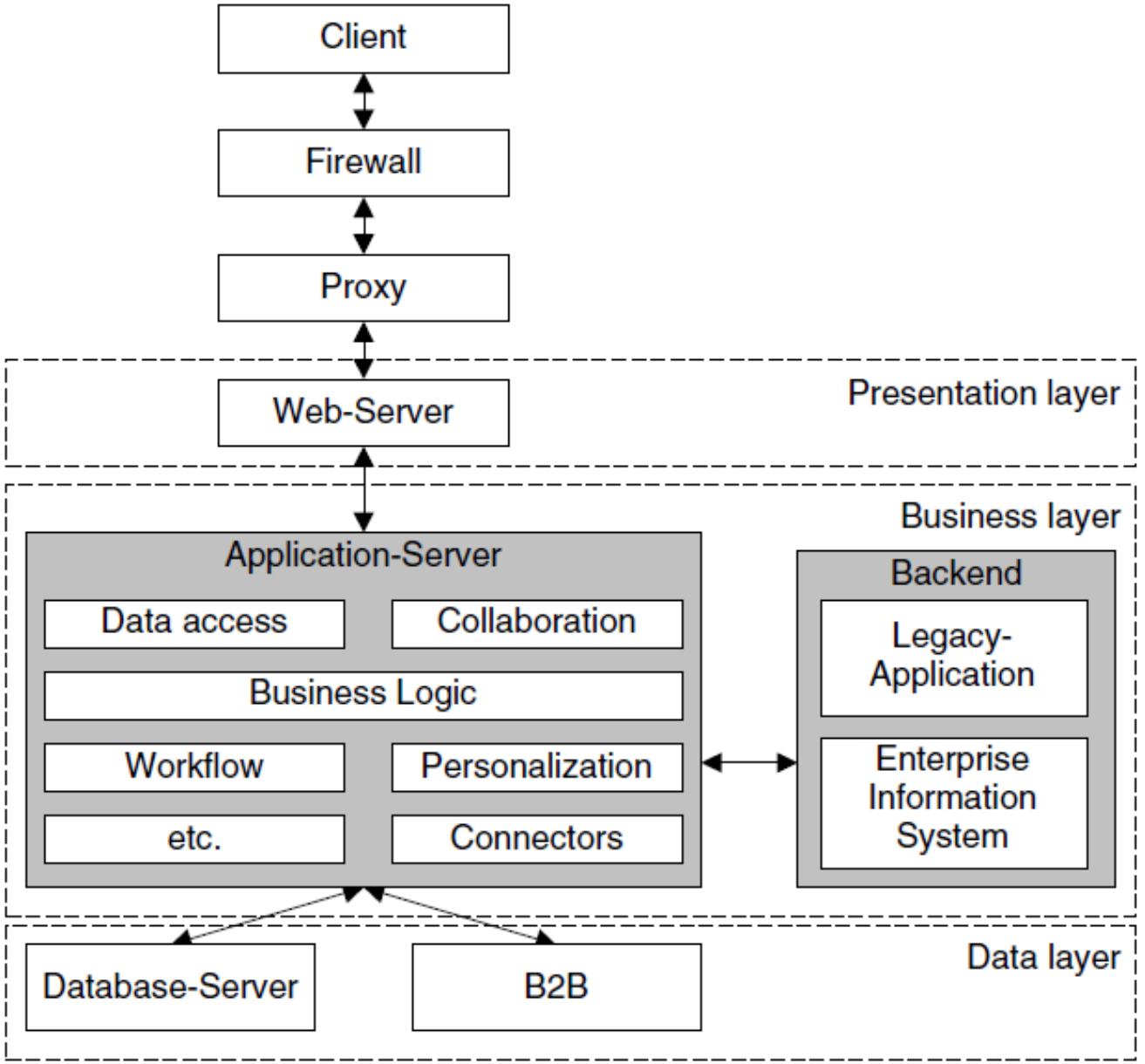
Layered Architectures

- **2-Layer Architectures - client/server architecture**



Layered Architectures

- **N-Layer Architectures**



Integration Architectures

- External or internal systems, e.g., existing applications, existing databases and interfaces to external business partners, can be integrated into Web applications on three levels:
  - The presentation level, the application logic level, and the content level.
- Integration architectures address integration aspects on the content level
- the application logic level - *Enterprise Application Integration* (*EAI*) architectures.

Categorizing Architectures

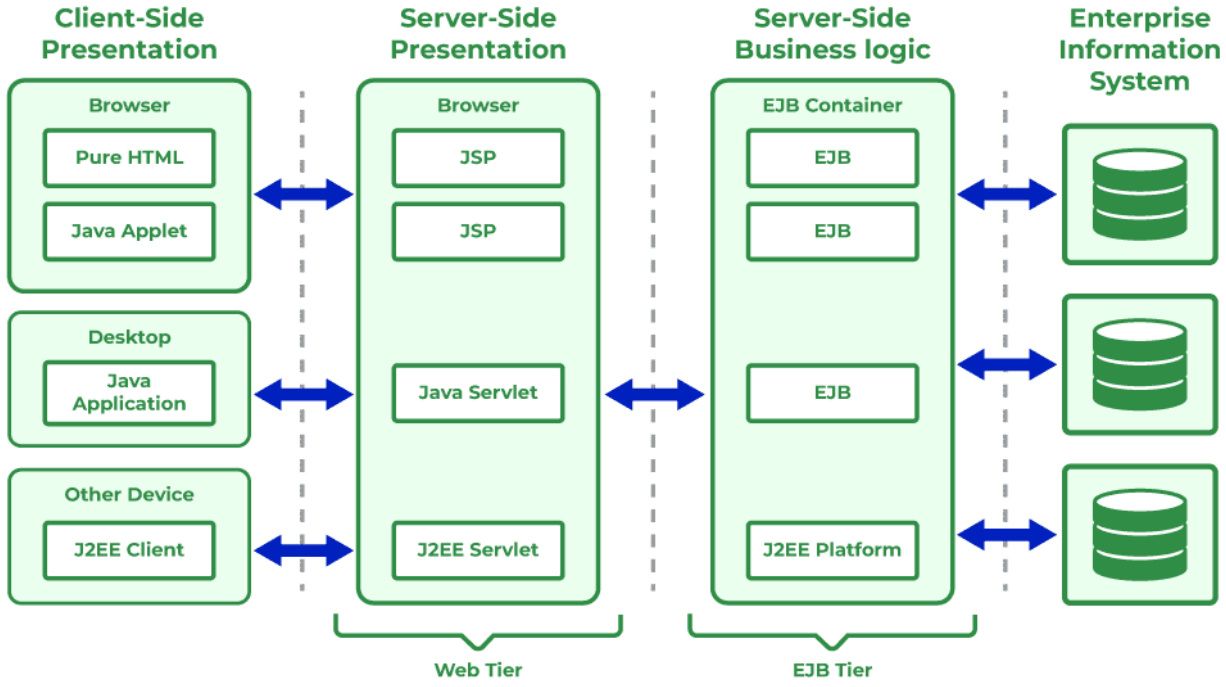
- **Layering aspect:**
- software systems are structured in several tiers to implement the principle of “separation of concerns” within a software system.
- Many frameworks in the field of distributed systems and Web applications are primarily structured by the layering aspect,
- e.g., J2EE (Sun Microsystems 2003a) Architectures used to integrate legacy systems, also referred to as Enterprise Application Integration (EAI), and portals also fall in this category.

Multitier Architecture

- **Clients**: It Refers to a program that requests service from a component.
- **Resource**: A resource is anything a component needs to provide a service.
- **Component**: It is part of a tier that consists of a collection of classes or a program that performs a function to provide a service.
- **Containers:** It is software that manages a component and provides system services to the component. A container handles persistence, resource management, security, threading, and other system-level services from components that are associated with the container. Example: APIs.

J2EE Multitier Architecture

- J2EE is a four-tier architecture that consists of:
- Client Tier/ Presentation Tier
- Web Tier
- Enterprise JavaBeans Tier/ Business Tier
- Enterprise Information Systems Tier



Categorizing Architectures

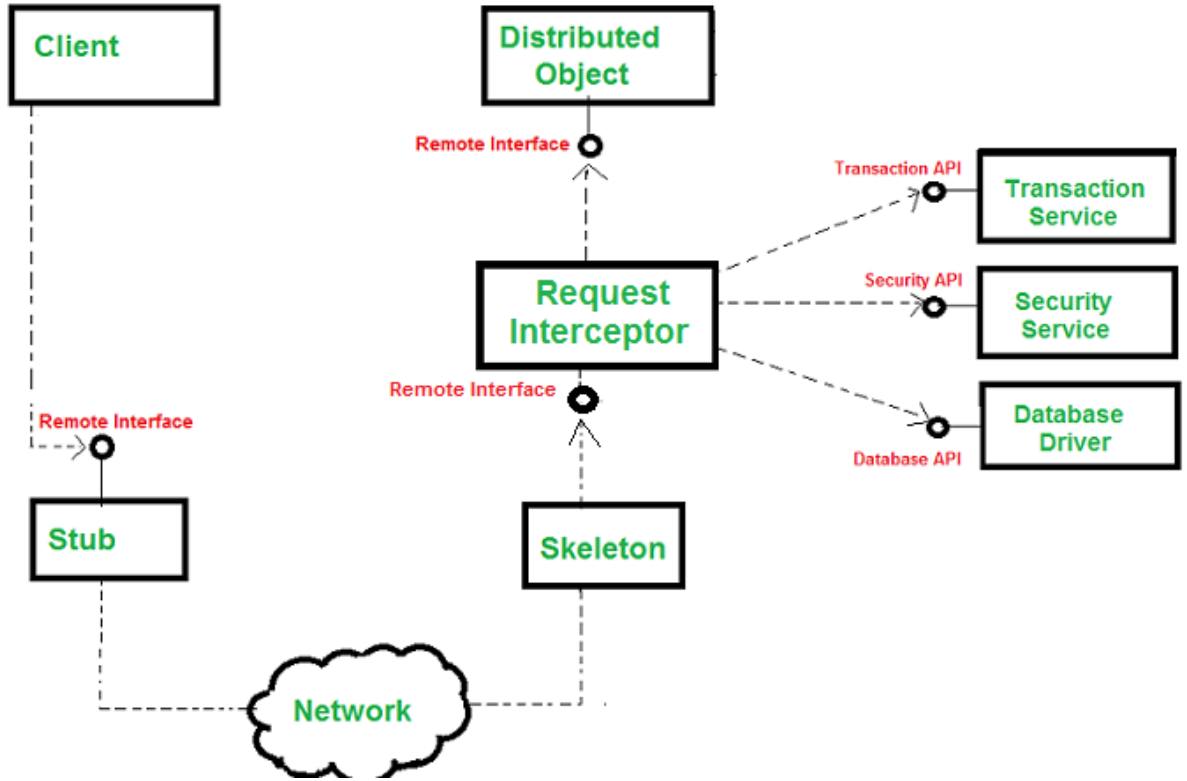
- Data aspect:
- Data can be structured or non-structured.
- Structured data follow a defined scheme like tables in a relational database or XMLstructures in a document.
- Non-structured data are multimedia contents, e.g., images, audio, and video, which typically do not follow an explicit scheme.

Categorizing Architectures

- The increasing distribution of software systems has led to the development of architectures and infrastructures addressing the distribution of data and messages:
- **Distributed Object Middleware (DOM):**
  - allows to access remote objects transparently.
  - It is based on the Remote Procedure Call (RPC) mechanism.
  - Some DOM systems also enable objects on different platforms to interact (e.g., CORBA).
  - Example: Microsoft's DCOM (Distributed Component Object Model), or EJB (Enterprise Java Beans) by Sun Microsystems.

Categorizing Architectures

- Distributed Objects in EJB(Enterprise Java Beans)



Categorizing Architectures

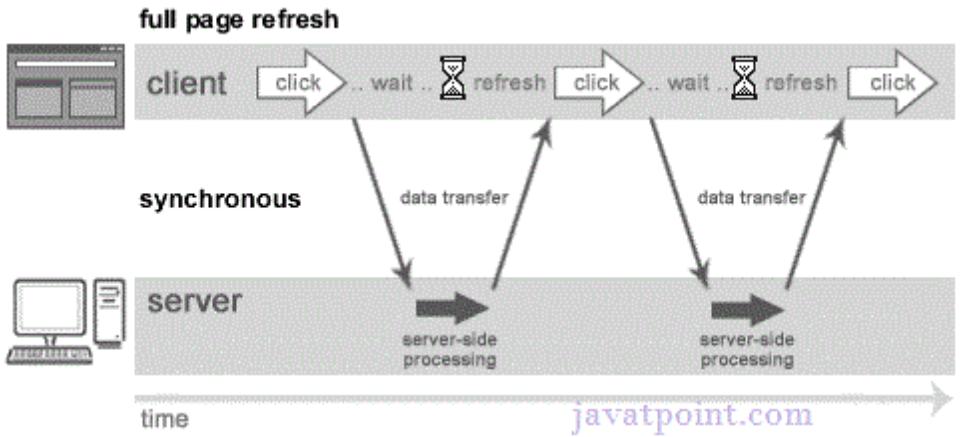
- **Virtual Shared Memory (VSM):**
  - The VSM model lets distributed processes access common data.
  - The processes themselves access a shared memory.
  - An appropriate middleware, transparent for the processes, is used to distribute the data.
  - This data can be “anywhere” in the system (hence “virtual”).
  - Examples: Corso

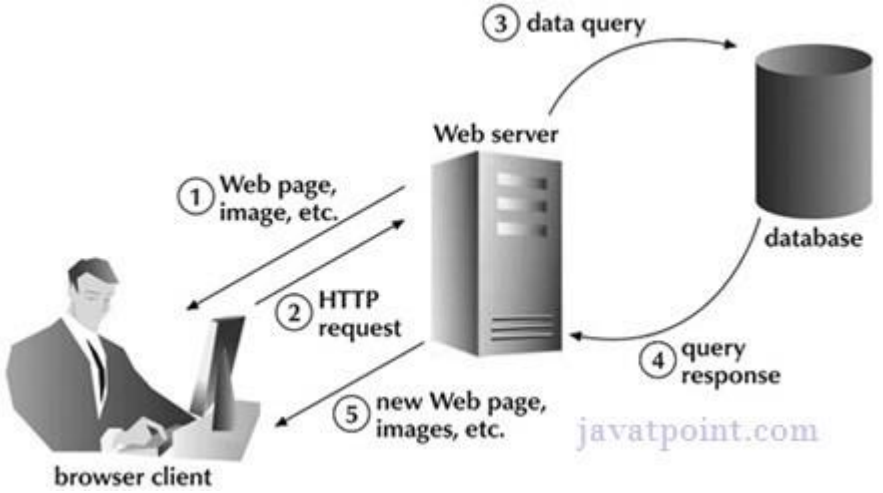
Categorizing Architectures

- Message Oriented Middleware (MOM):
- which is an infrastructure that allows communication and exchanges the data (messages).
- It involves the passing of data between applications using a communication channel that carries self-contained units of information (messages).
- In a MOM-based communication environment, messages are sent and received asynchronously.
  - Asynchronous communication differs from synchronous communication in that messages are sent to the receiver regardless of its status,
  - e.g., the receiver may not be available when the message is sent, i.e., he or she may be offline. MOM ensures that messages are delivered nevertheless.
  - Examples: Sun's JMS (Java Messaging Service) and Microsoft's MSMQ (Microsoft Message Queue).

Categorizing Architectures

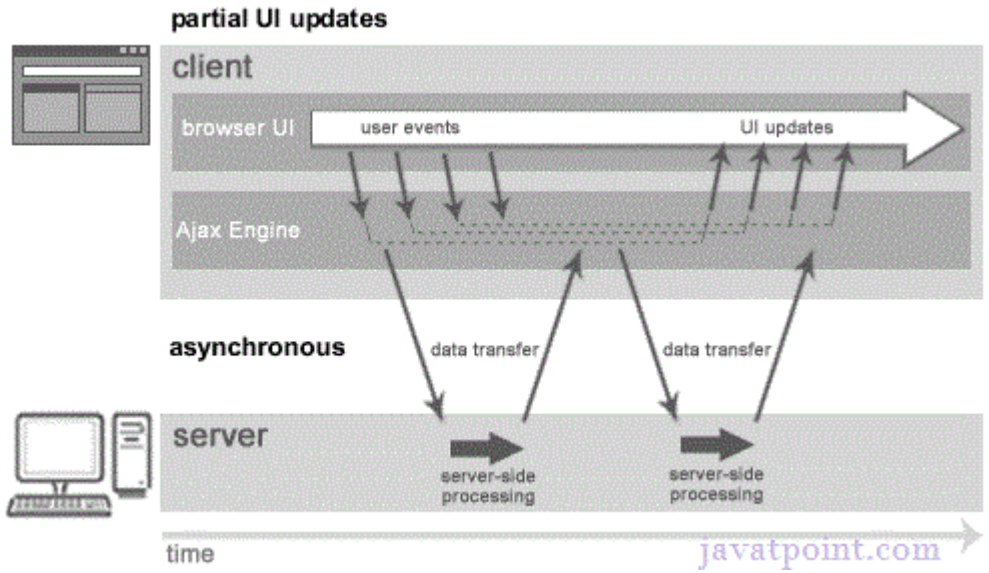
- Synchronous (Classic Web-Application Model)

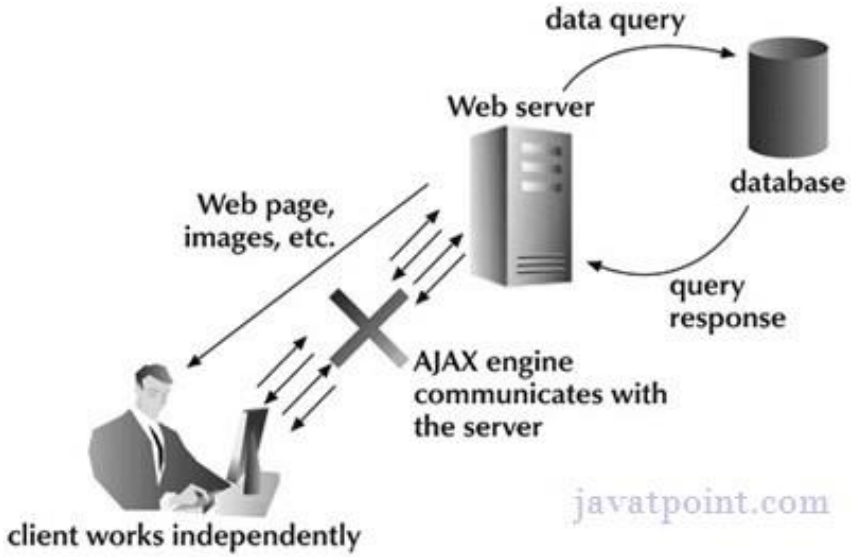




Categorizing Architectures

- Asynchronous (AJAX Web-Application Model)



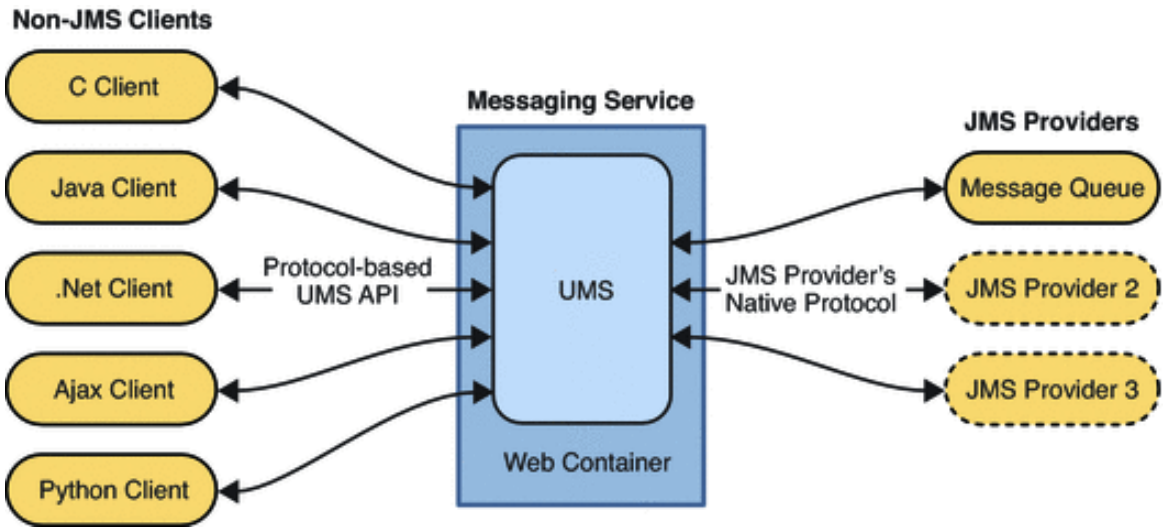


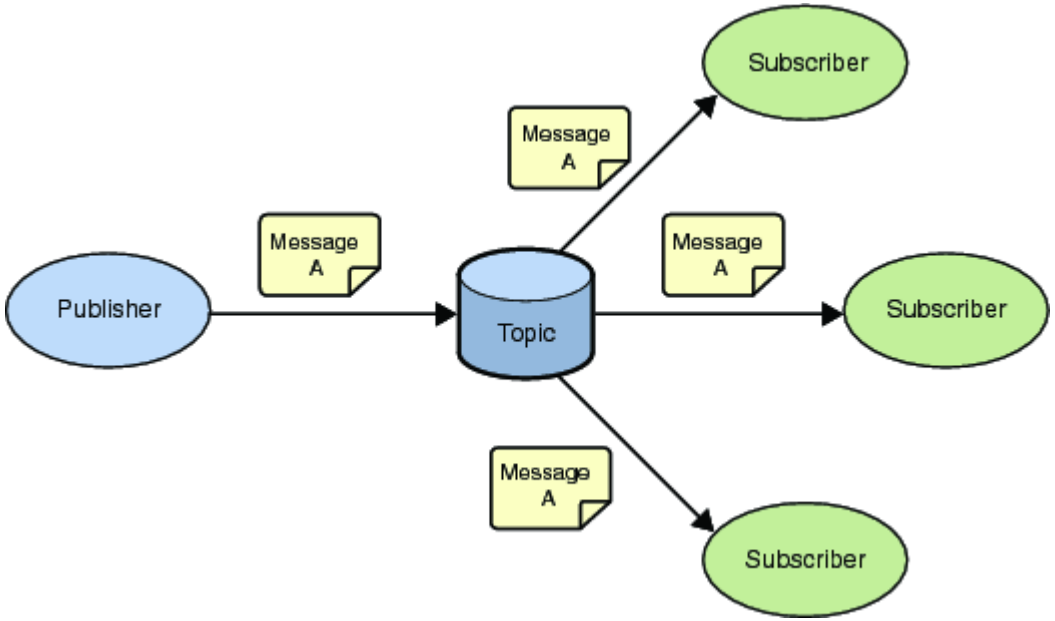
What is AJAX?

- Asynchronous JavaScript and XML (Ajax) refer to a group of technologies that are used to develop web applications.
- web pages appear more responsive since small packets of data are exchanged with the server and web pages are not reloaded each time that a user makes an input change.
- Ajax enables a web application user to interact with a web page without the interruption of constant web page reloading.
- Website interaction happens quickly with only portions of the page reloading and refreshing.
  - Ajax is made up of the following technologies: XHTML and CSS for presenting information.
    - Document Object Model (DOM) for dynamically interacting with and displaying the presented information.
    - XMLHttpRequest object to manipulate data asynchronously with the web server.
    - XML, HTML, and XSLT for data interchange and manipulation.
    - JavaScript for binding data requests and information display.

Categorizing Architectures

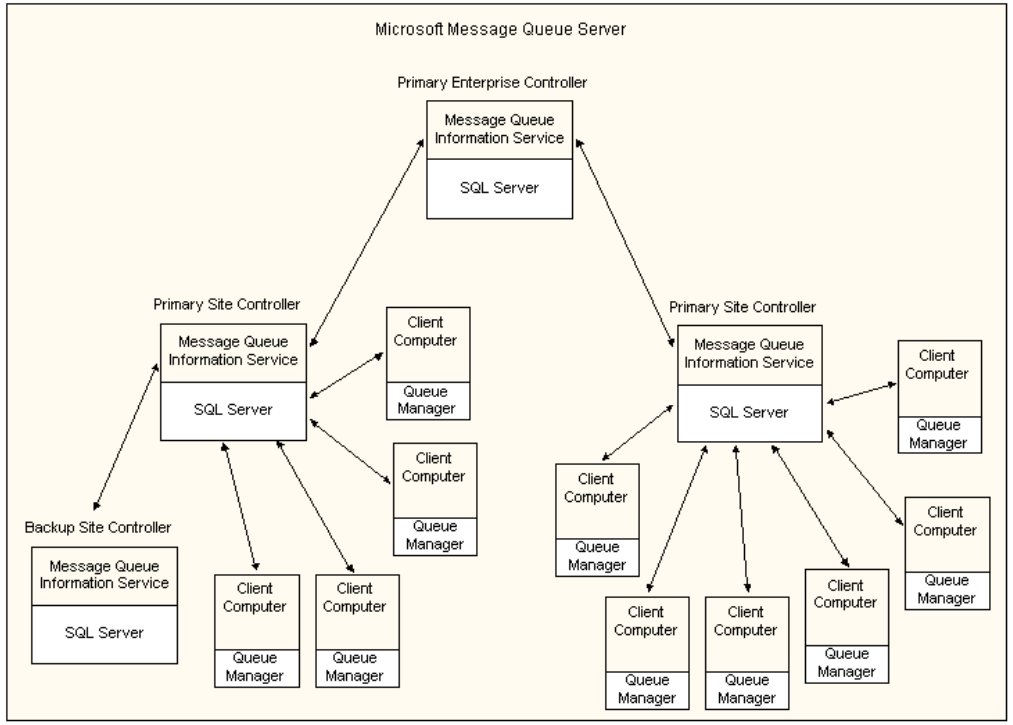
- Sun's JMS (Java Messaging Service)





Categorizing Architectures

- Microsoft's MSMQ is responsible for reliably delivering messages between applications inside and outside the enterprise.
- MSMQ ensures reliable delivery by placing messages that fail to reach their intended destination in a queue and then resending them once the destination is reachable.

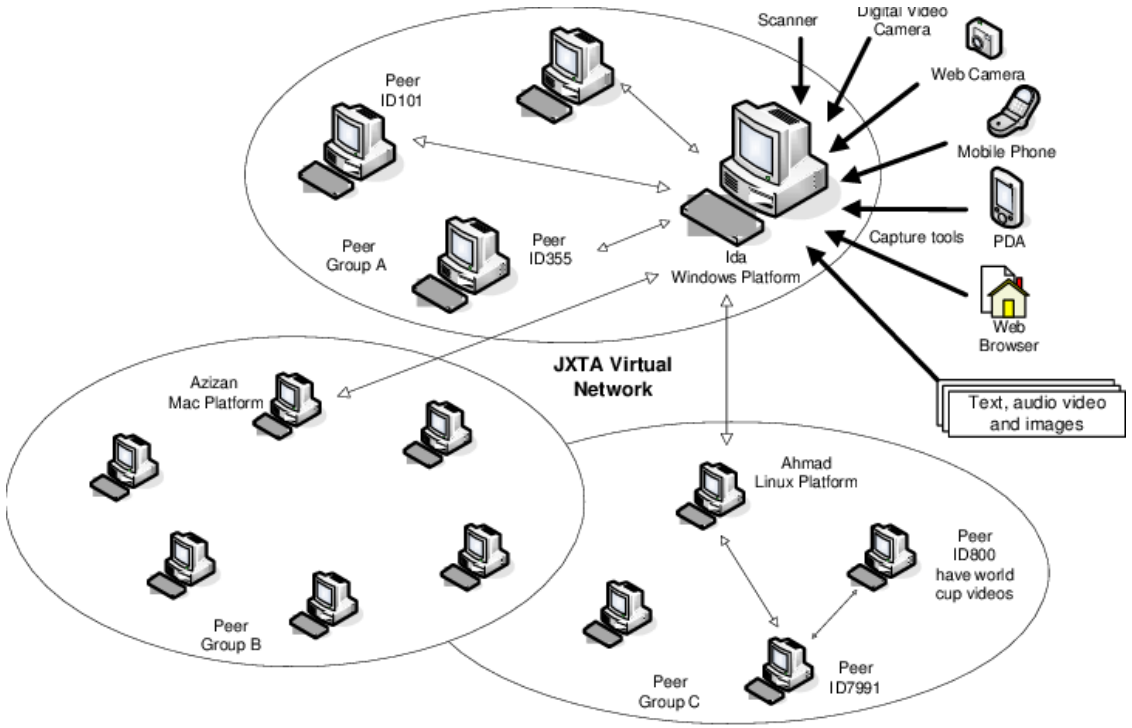


Categorizing Architectures

- Peer to Peer (P2P):
  - P2P stands for direct communication between two devices – the peers – in a system without using a server, i.e., they communicate over a point-to-point connection.
  - The peers are basically equal.
  - P2P systems describe how the devices in such a network communicate and how they can “discover” each other.
  - Example: JXTA

Categorizing Architectures

- The JXTA protocols were defined as a set of XML messages which allow any device connected to a network to exchange messages and collaborate independently of the underlying network topology.

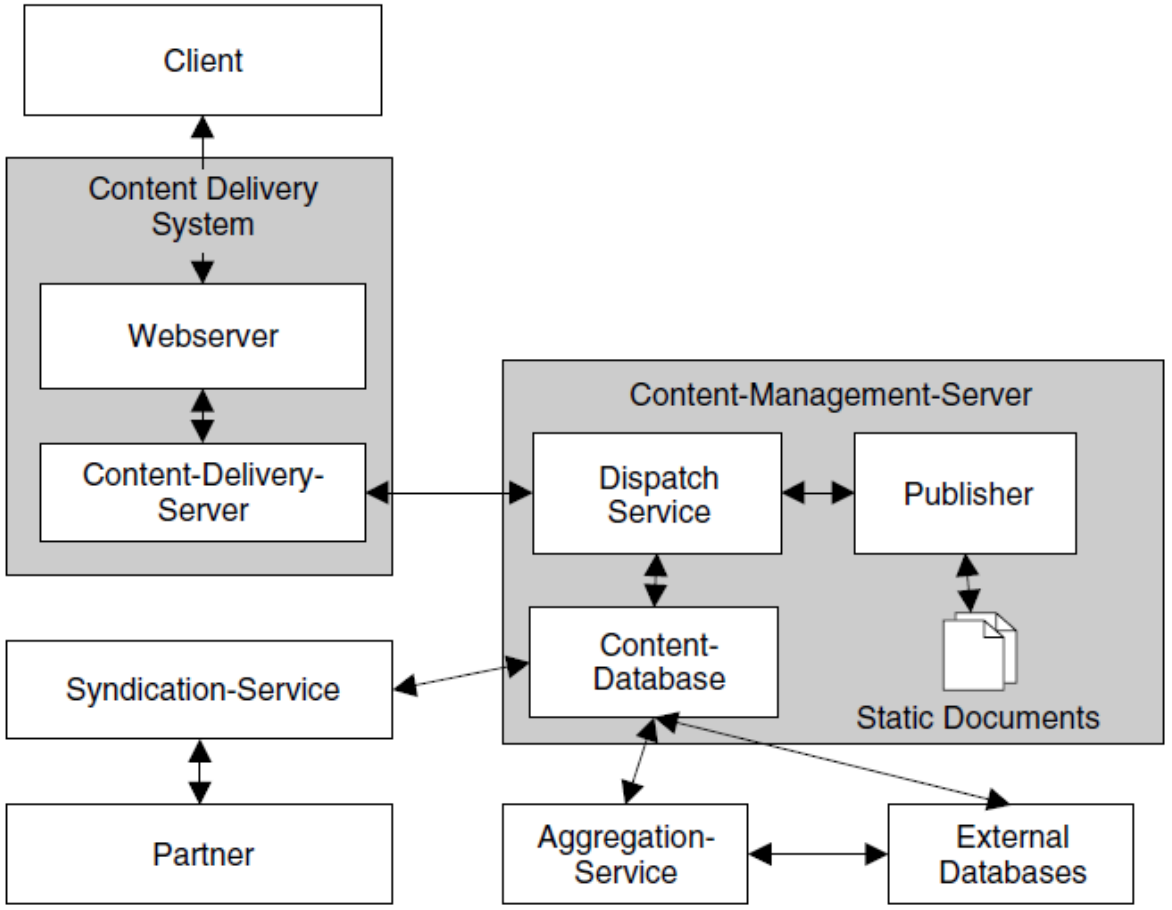


Categorizing Architectures

- Service Oriented Middleware (SOM):
  - SOM enhances DOM systems by the concept of services.
  - A service in this context is a number of objects and their behavior.
  - These objects use a defined interface to make a service available for other systems/services.
  - SOM defines communication protocols between services, and provides for location and migration-transparent access to services, thus supporting a simple integration of services beyond platform boundaries.
  - Example: Sun’s Jini system

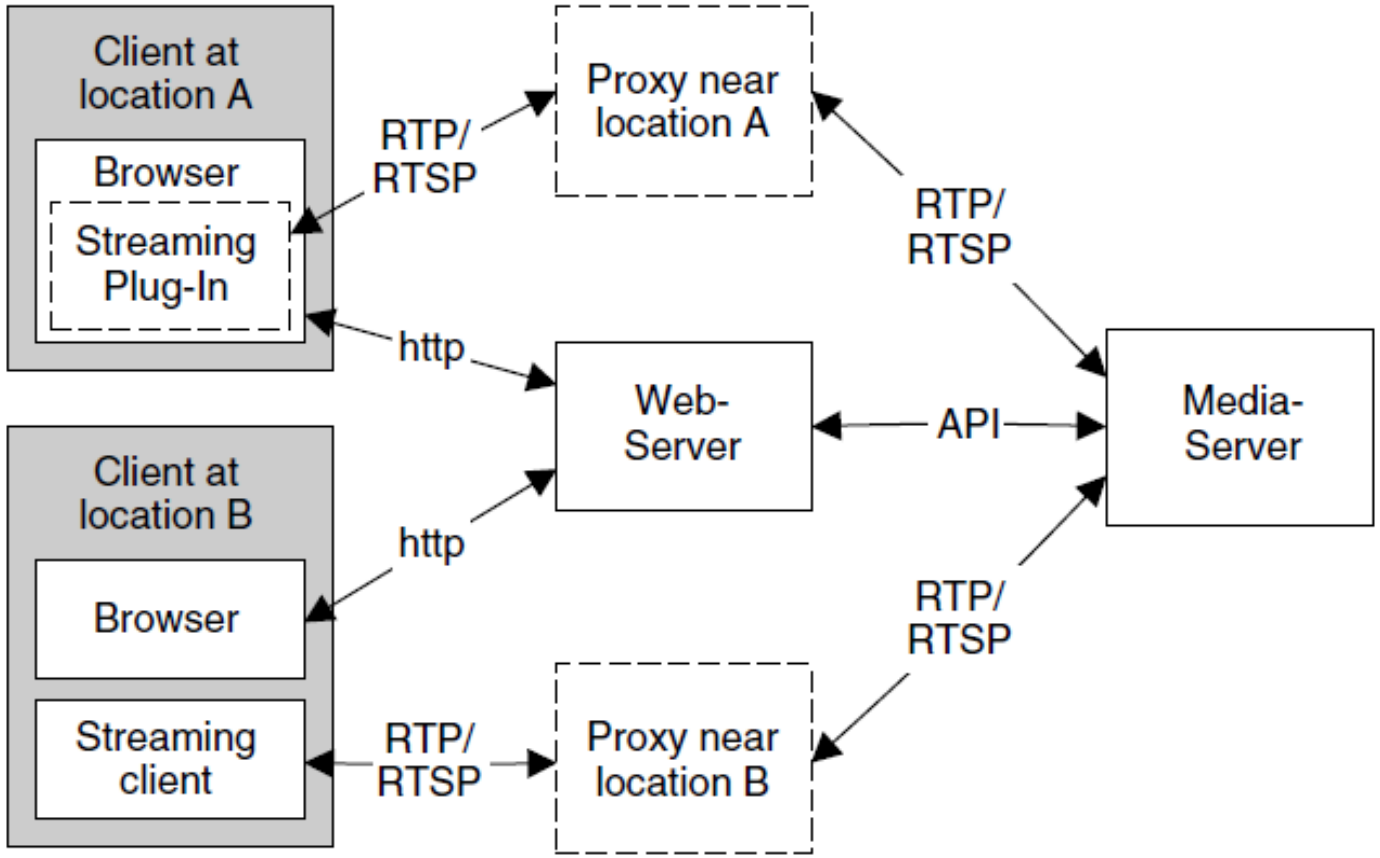
Architectures for Web Document Management

- A Web server receives a client request and forwards it to a *content delivery server*.



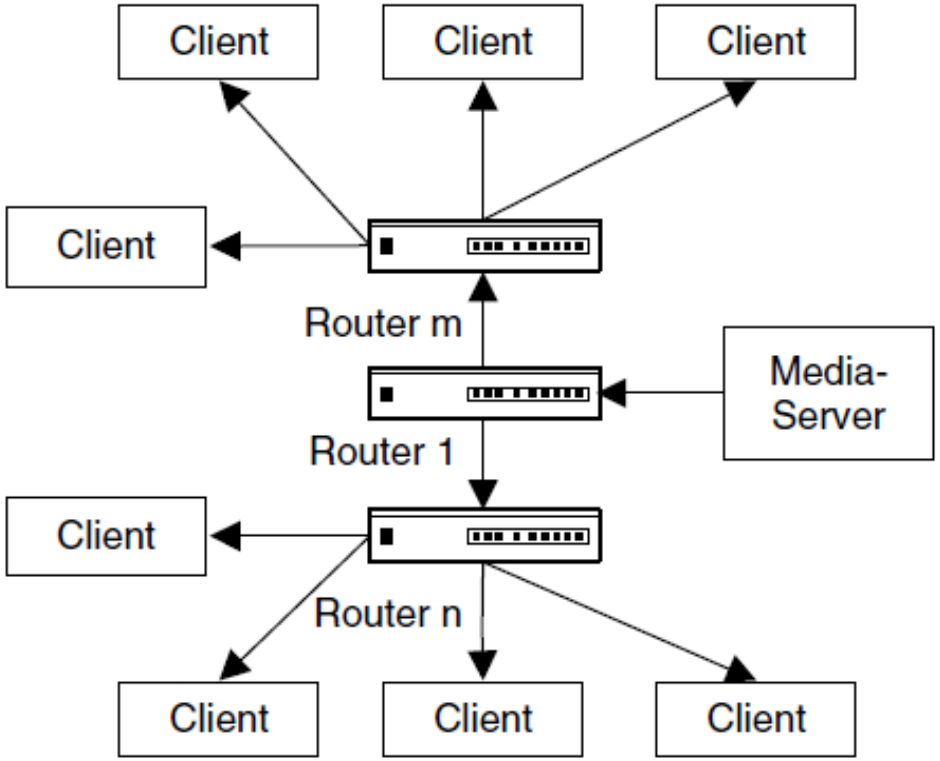
Architectures for Multimedia Data

- Streaming media architecture using point-to-point connections



Architectures for Multimedia Data

- Streaming media architecture using a broadcasting infrastructure



Reference

- Web Engineering classification and web development challenges
  - Web Engineering: Modelling and Implementing Web Applications by Gustavo Rossi
- Web Architecture-
  - Web Engineering - The Discipline of Systematic Development of Web Applications by Gerti Kappel