# The World Wide Web

A Case Study

主讲教师： 王灿

Email: wcan@zju.edu.cn

TA: 李奇平 liqiping1991@gmail.com

Course FTP: ftp://sa:sa@10.214.51.13

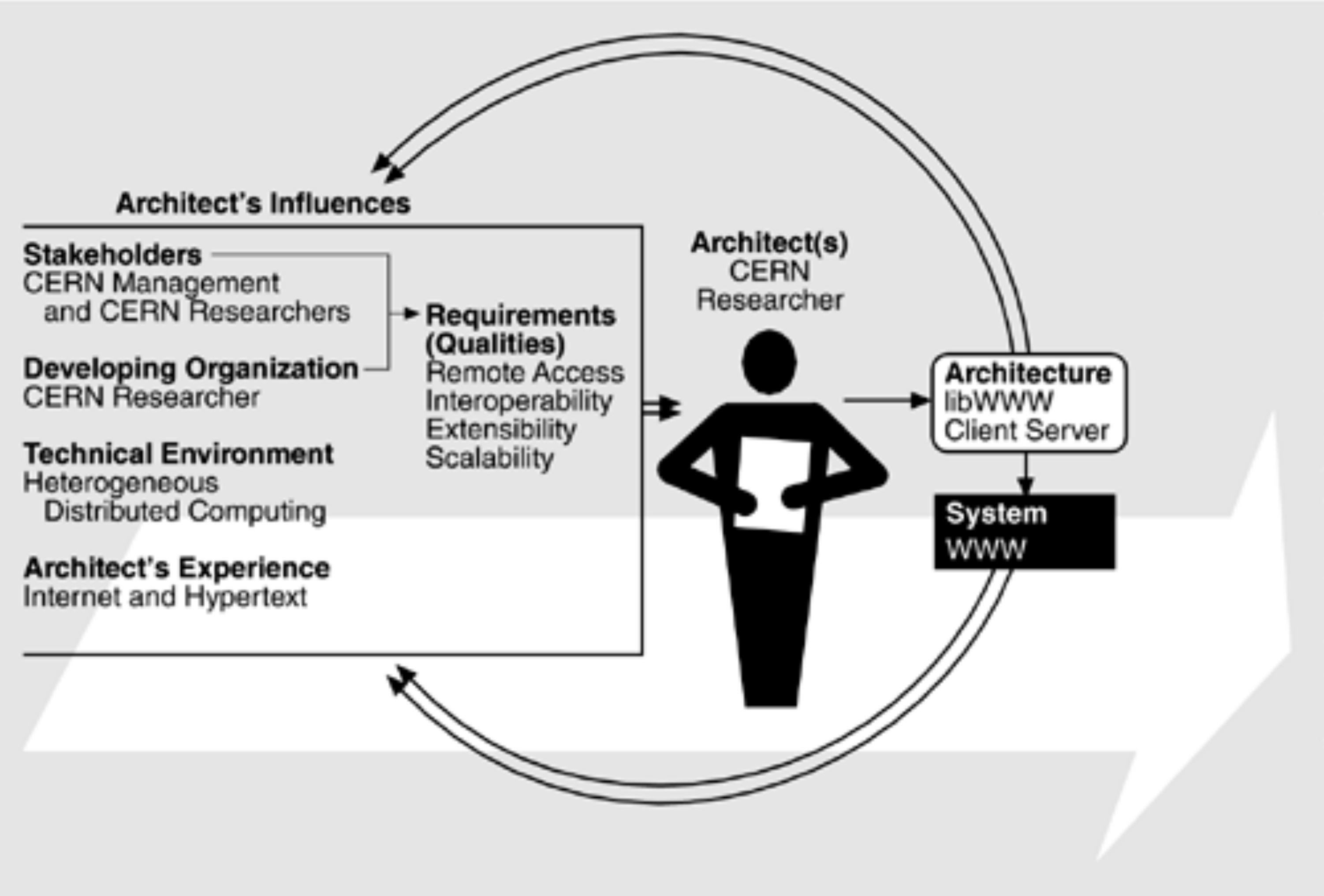
# What is the World Wide Web?

- A system of interlinked hypertext documents accessed via the Internet
- The largest Internet application
- An application platform for Web applications
- A virtual graphical layer of the Internet

# The Original Web

- 1989 Proposal from Tim Berners-Lee *“Information management: A Proposal”* at European Lab for Particle Physics (CERN)
  - Inspiration: the human “web” formed by the researchers at CERN.
  - Approved Oct., 1990
  - Named as the World Wide Web

# Requirements & Solution



Architect's Influences

**Stakeholders**
CERN Management
and CERN Researchers

**Developing Organization**
CERN Researcher

**Technical Environment**
Heterogeneous
Distributed Computing

**Architect's Experience**
Internet and Hypertext

**Requirements (Qualities)**
Remote Access
Interoperability
Extensibility
Scalability

**Architect(s)**
CERN
Researcher

**Architecture**
libWWW
Client Server

**System**
WWW

# Requirements and Qualities

- Business goals: *promoting interaction and allowing heterogeneous computing*
- Consequent quality goals:
  - Remote access
  - Interoperability
  - Extensibility
  - Scalability
- *What are the major design challenges given the existing techs at that time?*

# The Solution

- Berners-Lee's breakthrough was to marry hypertext to the Internet
  - Origins: 1940’s Vannevar Bush “hypertext”
  - HyperText Markup Language (HTML)  and Uniform Resource Locators (URLs)

- libWWW – the original library  that supported Web-based development using C/S architecture and layered design.

# The Original Requirements (1)

- Remote access across network
  - Any info shall be accessible from any machine on a CERN network
- Heterogeneity
  - Heterogeneous hardware and software platform
- Non-centralization
  - No constraints on the Web’s growth
- Access to existing data
- Ability for users to add data
  - Allow user to *publish* data
- Private Links

# The Original Requirements (2)

- *Bells and Whistles* – 24x80 display (graphics optional)
  - Scholars generally agree, that the turning point for WWW began with the introduction of the Mosaic Web browser in 1993, a graphical browser
- Data analysis
- Live links
- Other requirements explicitly mentioned but not dealt with in original proposal
  - Copyright enforcement
  - Data security (encryption of sensitive data SSL and HTTPS)

# Web Growth Statistics

| Date | Number of Web Sites | Percentage of .com Sites | Hosts per Web Server |
|---|---|---|---|
| 6/93 | 130 | 1.5 | 13,000 |
| 12/93 | 623 | 4.6 | 3,475 |
| 6/94 | 2,738 | 13.5 | 1,095 |
| 12/94 | 10,022 | 18.3 | 451 |
| 6/95 | 23,500 | 31.3 | 270 |
| 1/96 | 100,000 | 50 | 94 |
| 6/96 | 252,000 | 68 | 41 |
| 1/97 | 646,162 | 62.6 | 40 |
| 1/98 | 1,834,710 | | 16.2 |
| 1/99 | 4,062,280 | | 10.6 |
| 1/00 | 9,950,491 | | 7.3 |
| 1/01 | 27,585,719 | 54.68 | 4 |

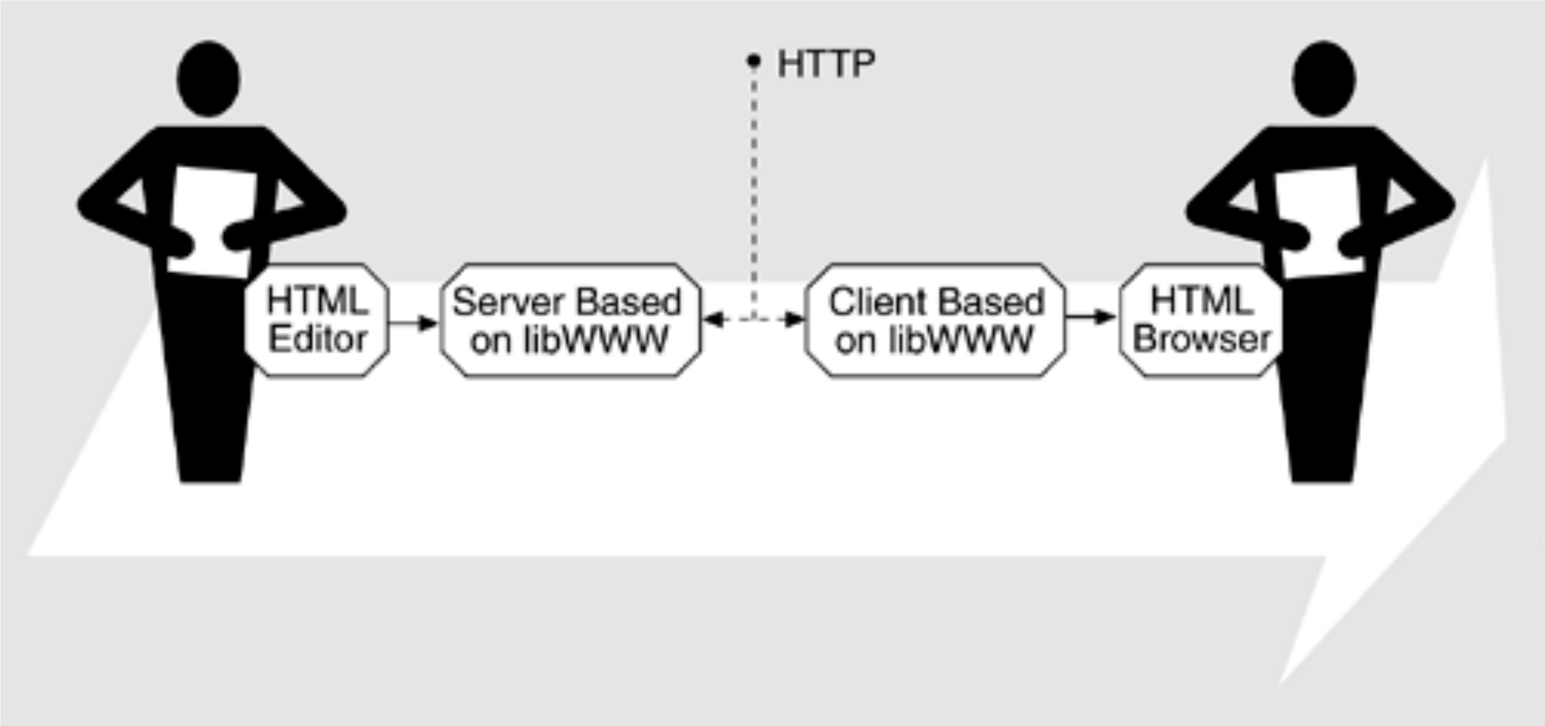
# Stay Simple
# ---The Key to the Explosive Growth

- Weak notion of central control
  - Unregulated style of interaction
- Features missing for hypertext systems design in the Web proposal
  - Controlling topology
  - Navigational techniques and user interface requirements
  - Having different types of links to express differing relationships among nodes
- Missing features in the original implementation: URL, HTML…

# Original Architectural Solution

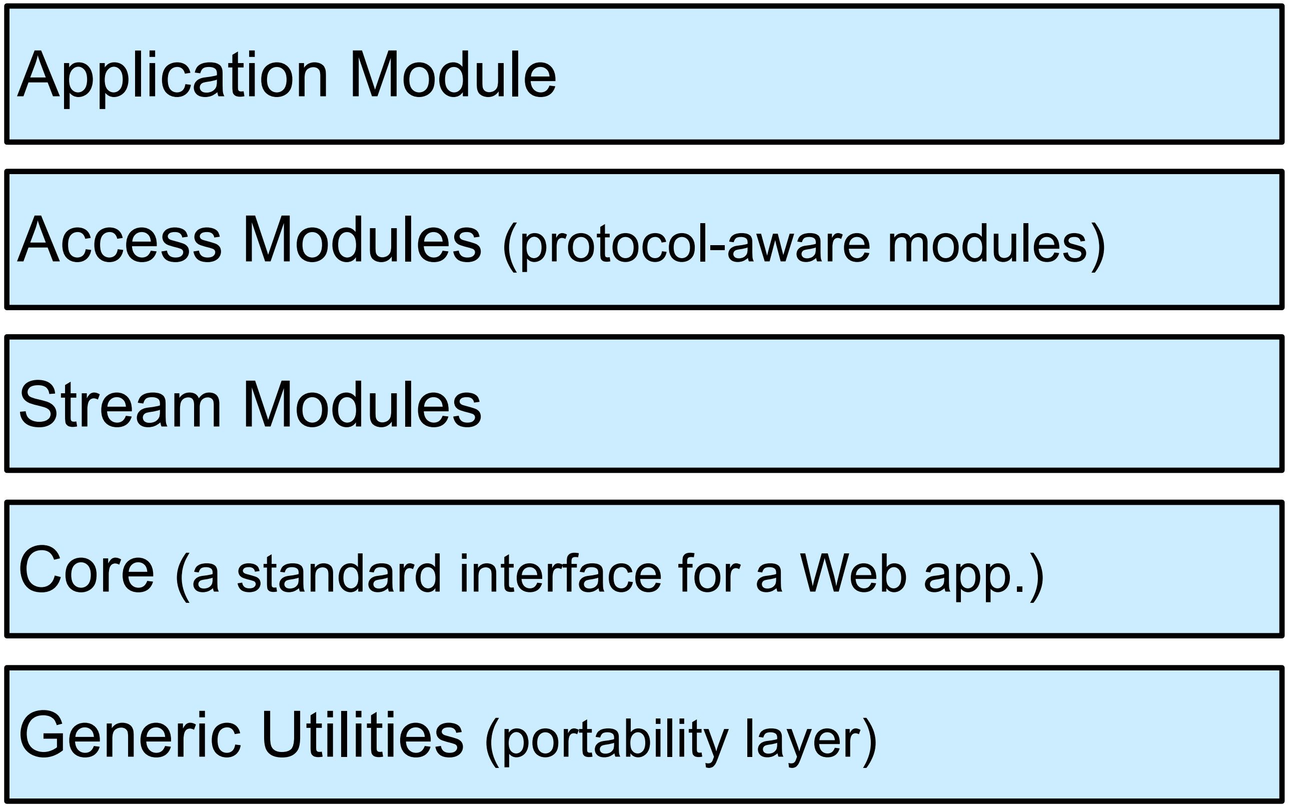
- Architectural approach first proposed at CERN and later by the World Wide Web Consortium (W3C):
  - The library libWWW hides all hardware, operating system and protocol dependencies
  - Using client-servers architecture
    - Client side “browser” that knows how to display HTML
    - Server – that knows how to interpret requests for pages and return the HTML pages
  - HTTP protocol – network protocol for transferring data packets (built on top of TCP/IP)

# Content Producers←→Consumers



libWWW is a library of software for creating applications that run on either the client or the server

# A Layered View of libWWW



Application Module

Access Modules (protocol-aware modules)

Stream Modules

Core (a standard interface for a Web app.)

Generic Utilities (portability layer)

# Generic Utilities Layer

- A portability layer
  - Masks platform difference
  - All higher layer can then be platform independent
- Functionalities provided in this layer
  - Network management
  - String manipulation
  - Data type extension

# Core Layer

- Containing the skeletal functionality of a **Web** application
  - Network access
  - Data management and parsing
  - Logging
- Providing a standard interface for a Web application
  - Actual functionalities provided by plug-in modules and call-out functions

# Stream Layer

- The stream layer provides the abstraction of a stream of data used by all data transported between the application and the network.

# The Access Layer

- Providing a set of network-protocol-aware modules
  - HTTP, NNTP, WAIS, TELNET, Gopher, rlogin etc.
  - Protocols are built upon the abstractions of the lower layers

# Application Layer

- A set of functionality useful for writing applications
  - caching
  - logging
  - registering proxy servers
  - gateways
  - history maintenance, etc.

# Lessons From libWWW (1)

- Allowing the features of Web-based apps to grow has driven many decision in libWWW and led to the following lessons:
  - Formalized application programming interfaces (APIs) are required
    - Supporting application development in a variety of platforms and languages
  - Layered functionality and APIs
    - Providing access to different levels of service abstraction

# Lessons From libWWW (2)

- The library must support a dynamic, open-ended set of features
  - Including run-time replacement of features
- Processes built on top of library must be thread safe to allow concurrent functionality
  - E.g. downloading large file over a slow communication link may take a considerable amount of time

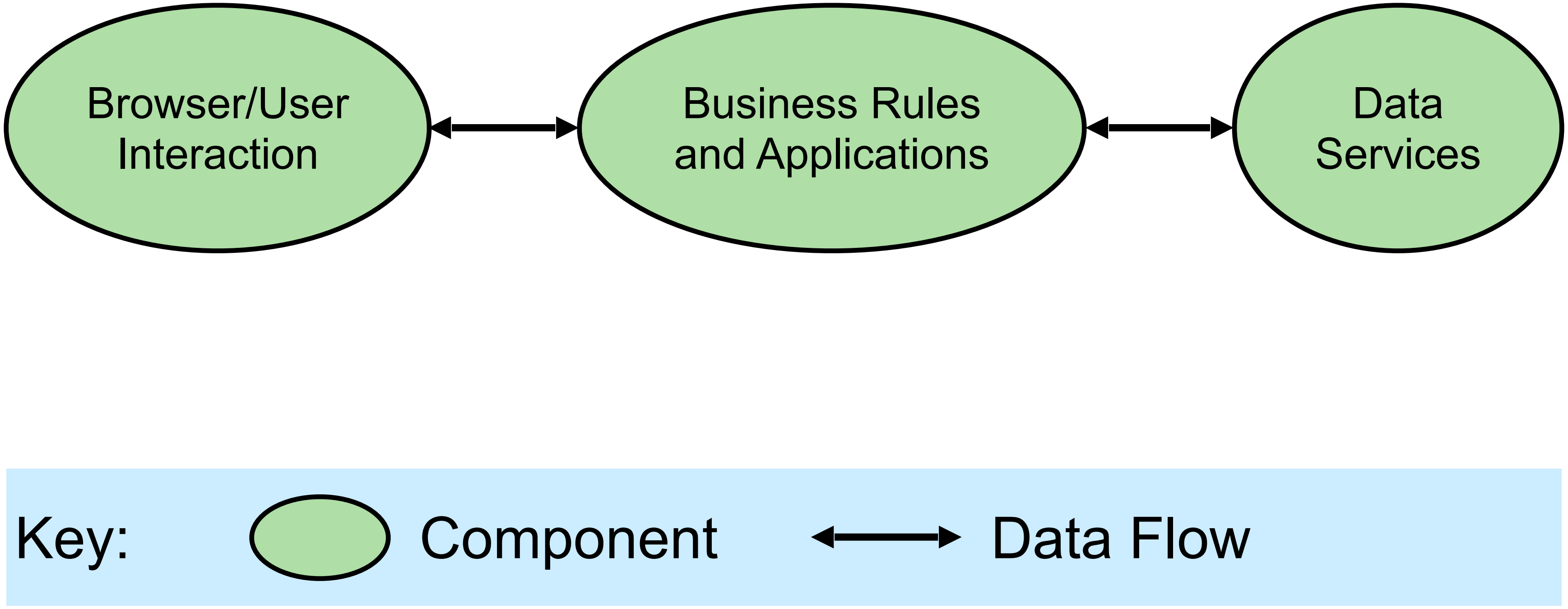
# How the WWW Achieved Its Initial Quality Goals

| Goal | How Achieved | Tactics Used |
|---|---|---|
| Remote access | Build Web on top of Internet | Adherence to defined protocols |
| Interoperability | Use libWWW to mask platform details | Abstract Common Services<br>Hide Information (Encapsulate) |
| Extensibility of software | Isolate protocol and data type extensions in libWWW;<br>Plug-ins | Abstract Common Services, Hide Information, replace components (Defer binding), config files (Defer binding) |
| Extensibility of data | Make each data item independent except for references it controls | Restrict Dependencies |
| Scalability | Use client-server architecture and keep references to other data local to the referring data location | Introduce concurrency<br>Reduce computational overhead |

# Evolution of Web-Based E-Commerce

- “Web of documents” VS. “Web of data”
- New requirements for e-commerce:
  - High performance
  - High availability
  - Scalability
  - Security
  - Modifiability
- System architecture versus software architecture
  - Frequently use COTS as system components

# A Simple E-commerce Reference Architecture



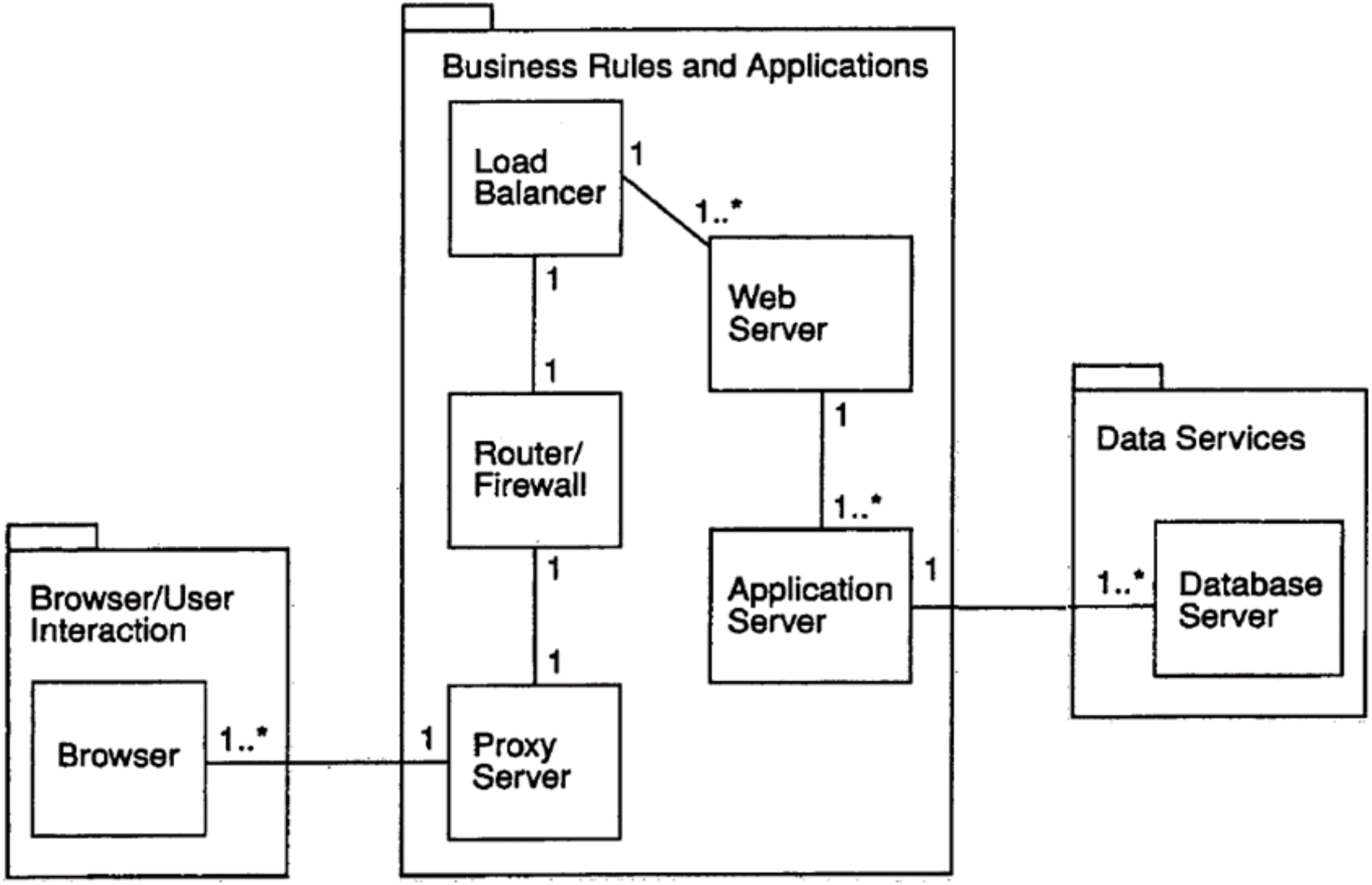
Browser/User Interaction

Business Rules and Applications

Data Services

Key: Component ⟷ Data Flow

# A Typical E-commerce System



# Achieving Qualities (1)

- Web browsers for modifiability
  - The browser-based user interface is not hardwired, but specified via HTML
    - User interface is naturally separated from the rest of the application
- HTTPS for security
  - Using port 443 instead of the standard port 80 to request TCP/IP services in an encrypted form
  - Using a 128-bit public/private key pair to encrypt the data
    - Sensitive data such as credit card or ID numbers

# Achieving Qualities (2)

- Proxy Servers for Performance
  - Caching frequently accessed Web pages so that users may retrieve them without having to access the Web site
  - Typically located close to the users
- Routers and firewalls for security
  - Routers may perform NAT transformation
  - *Packet filters* examine the TCP and IP headers to detect bad behavior
  - *Application proxy* to filter traffic based on known patterns of behavior

# Achieving Qualities (3)

- Load balancing for performance, scalability and availability
  - Distributing the "load" among a pool of computers running Web servers
  - ***Horizontal scaling*** (adding more instances of a given resource)
  - Active redundancy by monitoring the aliveness of each of its computers

- Web servers for performance
  - Multithread
  - Multithread (***Vertical scaling***, i.e. adding more powerful instances of a given resource)

# Achieving Qualities (4)

- Application servers for modifiability, performance and scalability
  - Separating business logic from "fat client" and DB
  - Load balancing, horizontal scaling
- Databases for performance, scalability and availability
  - Multi-layer architecture
  - Internal replication

# How the E-Commerce Architecture achieves its Quality Goals

| Goal | How Achieved | Tactics |
|---|---|---|
| High Performance | Load balancing, proxy servers | Introduce concurrency, increase resources, multiple copies |
| High Availability | Redundant processors, networks, databases, load balancing | Active redundancy, transactions, introduce concurrency |
| Scalability | Allow for horizontal and vertical scaling | Abstract common services, introduce concurrency |
| Security | Firewalls, public/private key encryption, NAT | Limit access, integrity, limit exposure |
| Modifiability | Separation of functionality: (browser), DB design, business logic (3 tiers) | Abstract common services, intermediary, interface stability |

# Reading Assignment

- Read chapter 23: Economic Analysis of Architectures. The next lecture will present CBAM