Enterprise Software Architecture & Design

UNIVERSITY OF Waterloo

Characteristics

Servers

- application server, web server, proxy servers etc.

Clients

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

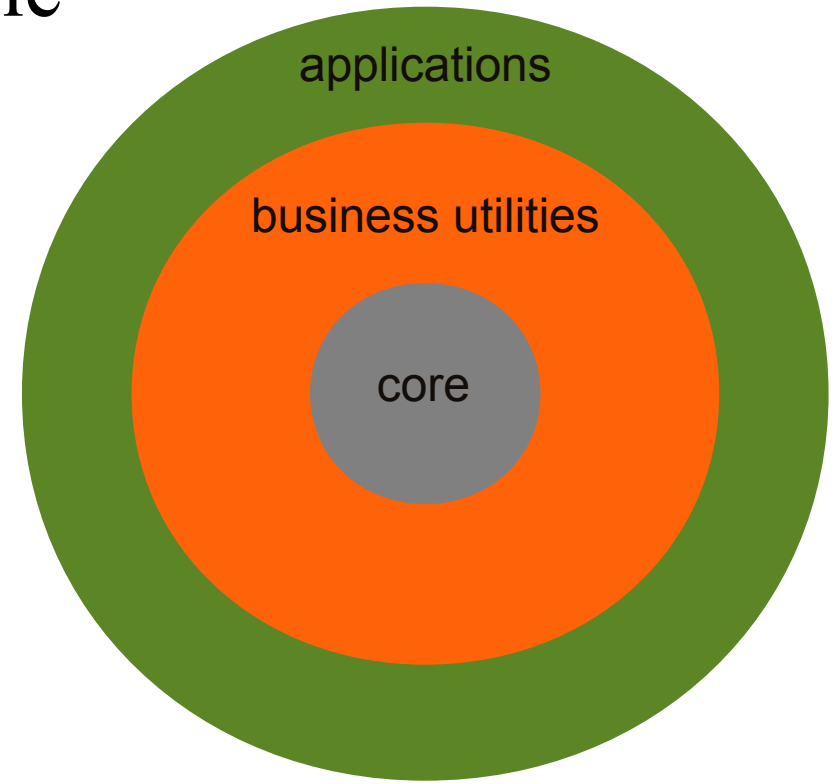
Characteristics

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

Architectural Style

Layered style



Are you sure?

Architectural Style

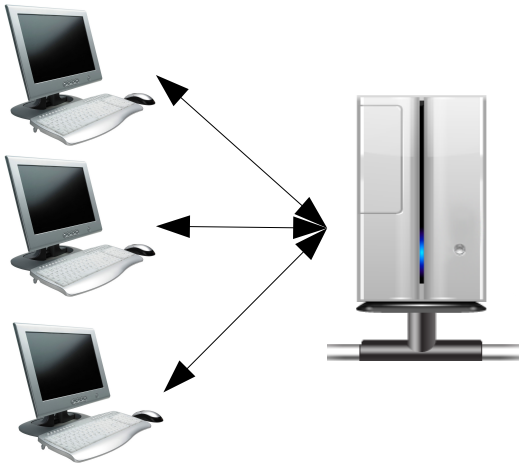
Tiered style

- 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

Architectural Style

Client-Server style

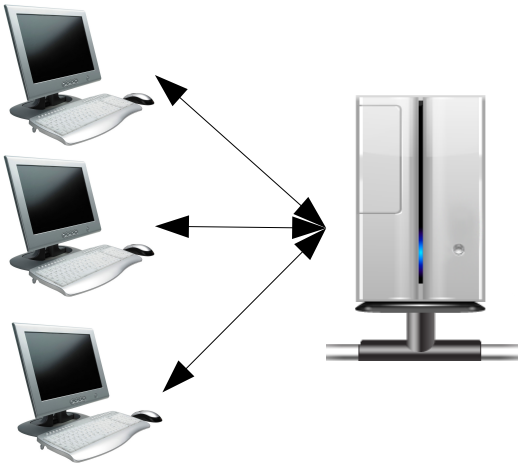
- distributed clients
  - thick & thin
  - isolated from each other
- centralized servers
  - computationally powerful
  - one server to support many clients



Architectural Style

Client-Server style

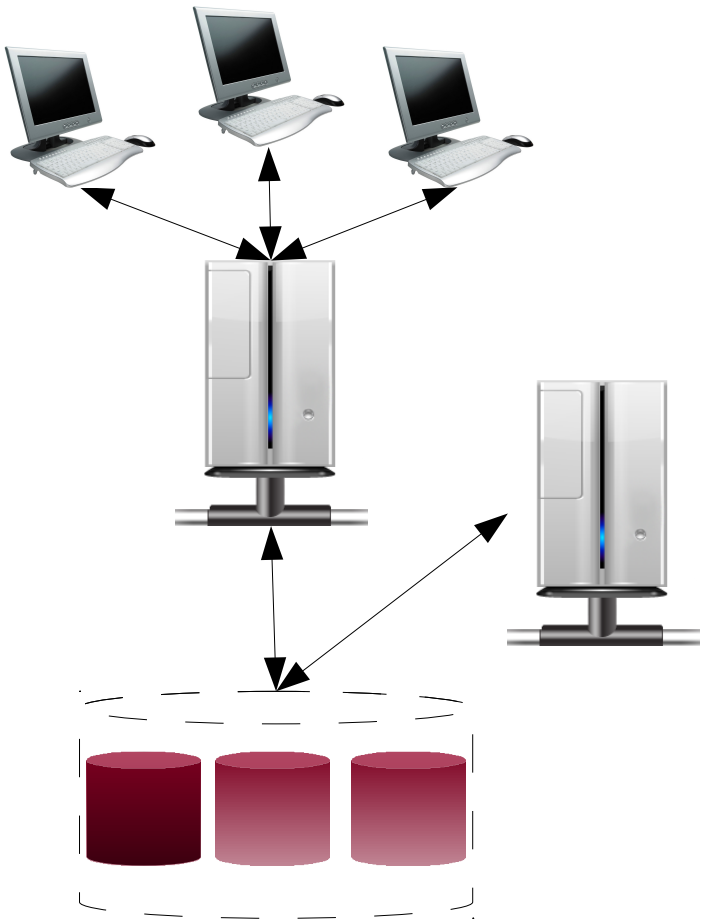
- observations
  - main functionality processed at the central server
  - user interface at each client
  - flows
    - data flows from server to client
    - control flows from client to server
  - *did some body say data?*



Architectural Style

Repository Style

- central repository
  - multiple data-sources
  - generally database type
- data is shared across
  - clients
  - applications
- data is dynamic
  - in enterprise applications



Functional Concerns

application

- collection of business functionality
- generally divided over two tiers

data

- transactional
  - transaction – generally a single operation
  - ACID
    - atomicity – all or nothing
    - consistency – from one consistent state to another consistent state
    - isolation – interaction of other operations with the modified data
    - durability – data after a successful transaction is never lost

University of Waterloo

Non-functional Concerns

Mostly honoured

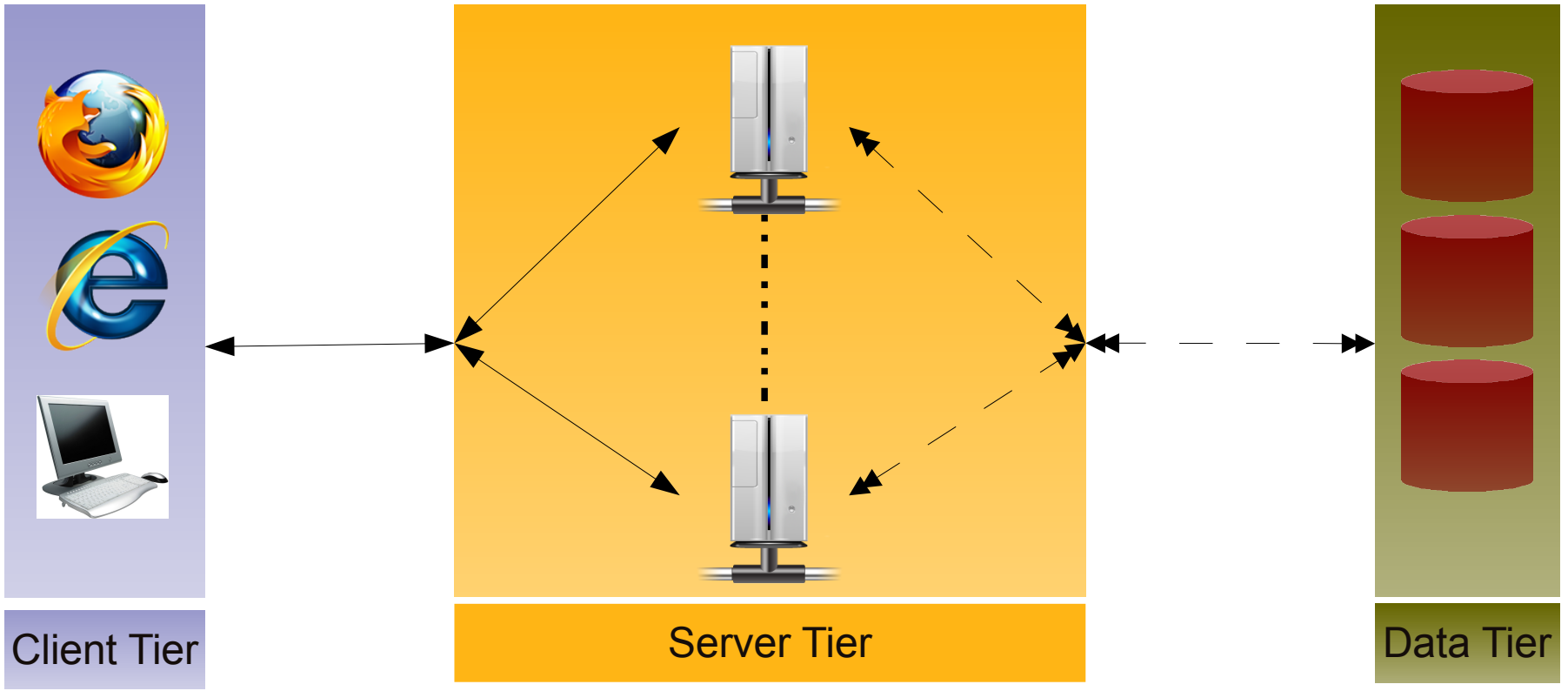
- concurrency
- availability
- security
- performance
- fault-tolerance
- application distribution & deployment
- evolution
- re-usability

Non-functional Concerns

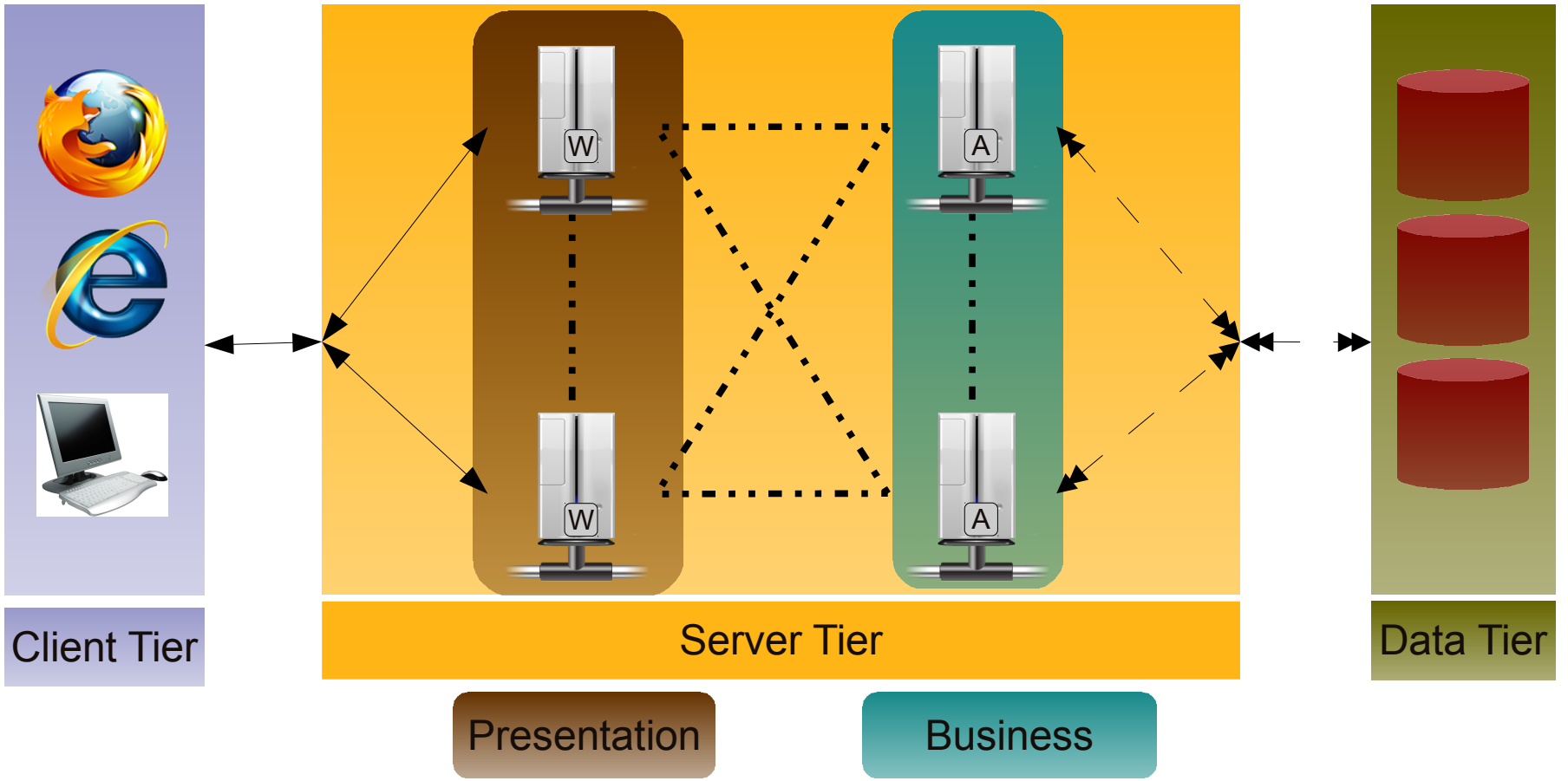
Most commonly violated

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

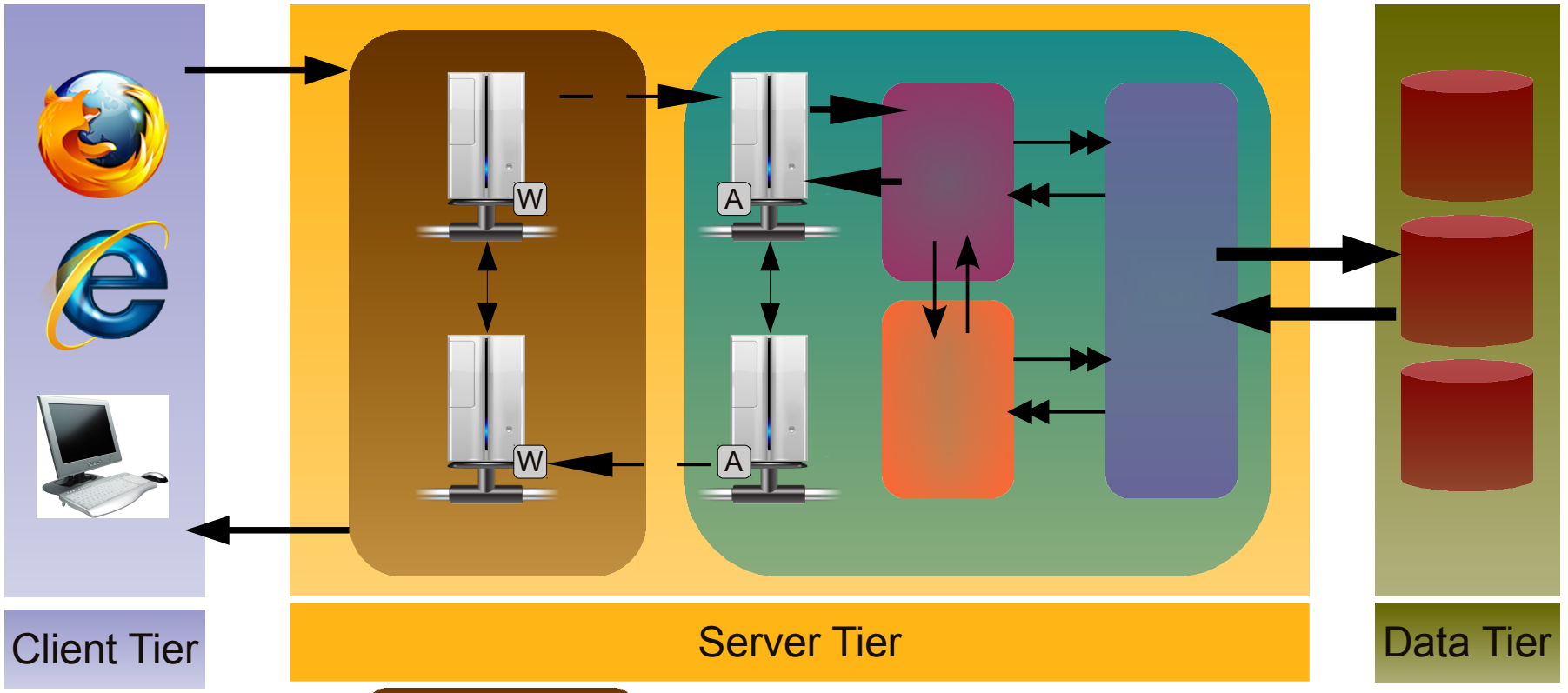
Putting it all Together

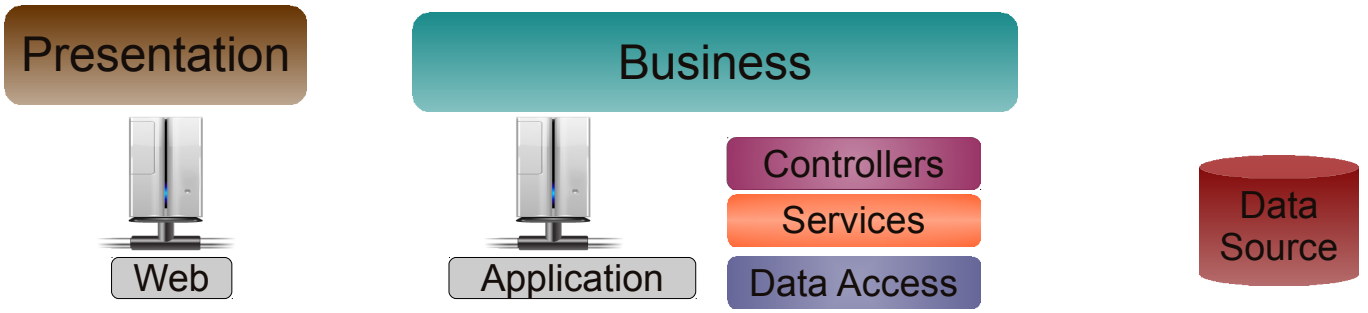


Putting it all Together



Putting it all Together





Web-based Enterprise Apps

Why web applications?

- what non-functional requirements are we solving?

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

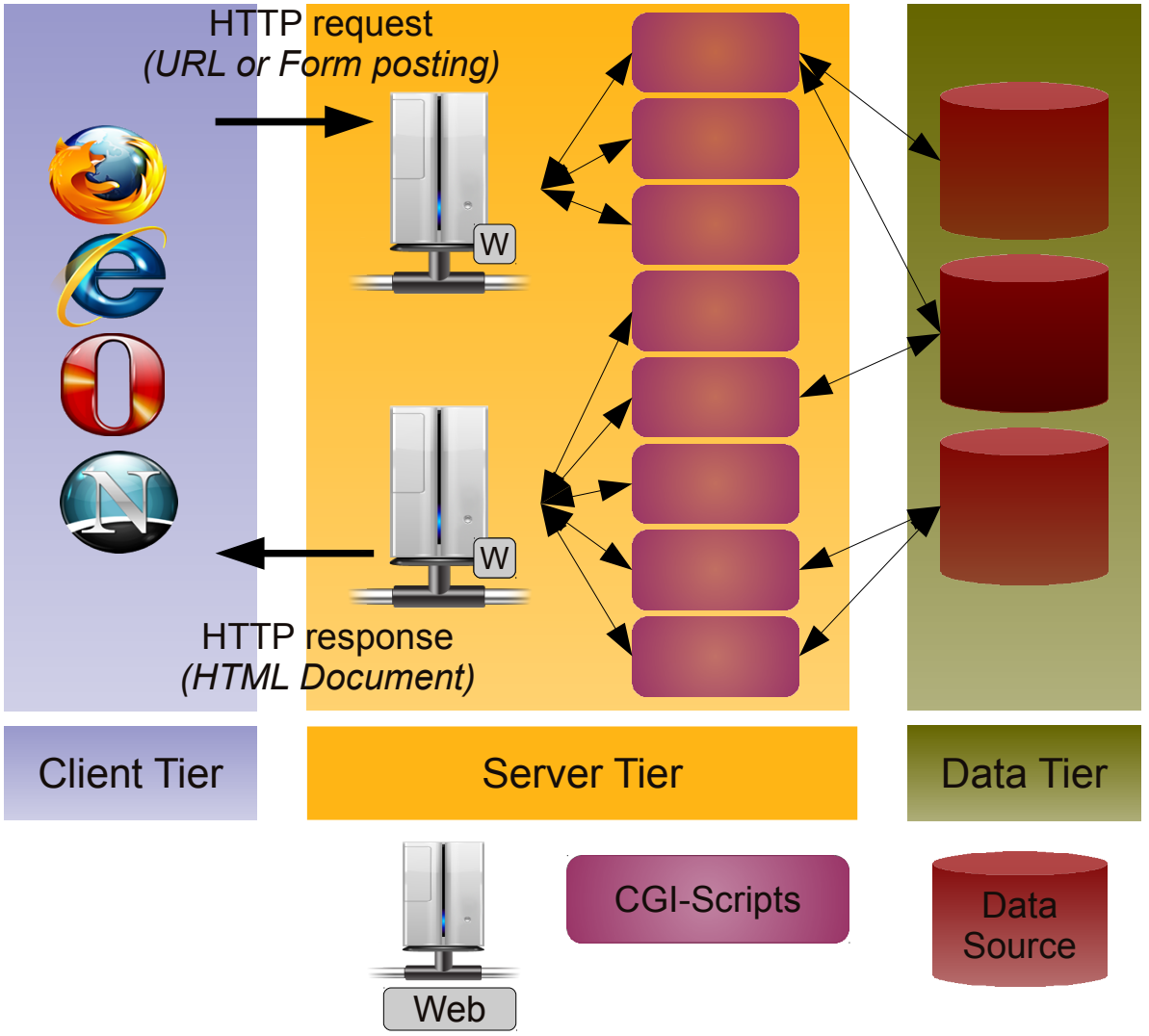
cost
ease of use
interoperability
portability
throughput

Web-based Enterprise Apps

Key attributes

- thin clients – web browsers
  - computationally challenged
- user interface – HTML, javascript, css
  - simple & static
  - resides at client tier
- communication
  - synchronous request response
  - HTTP over TCP/IP

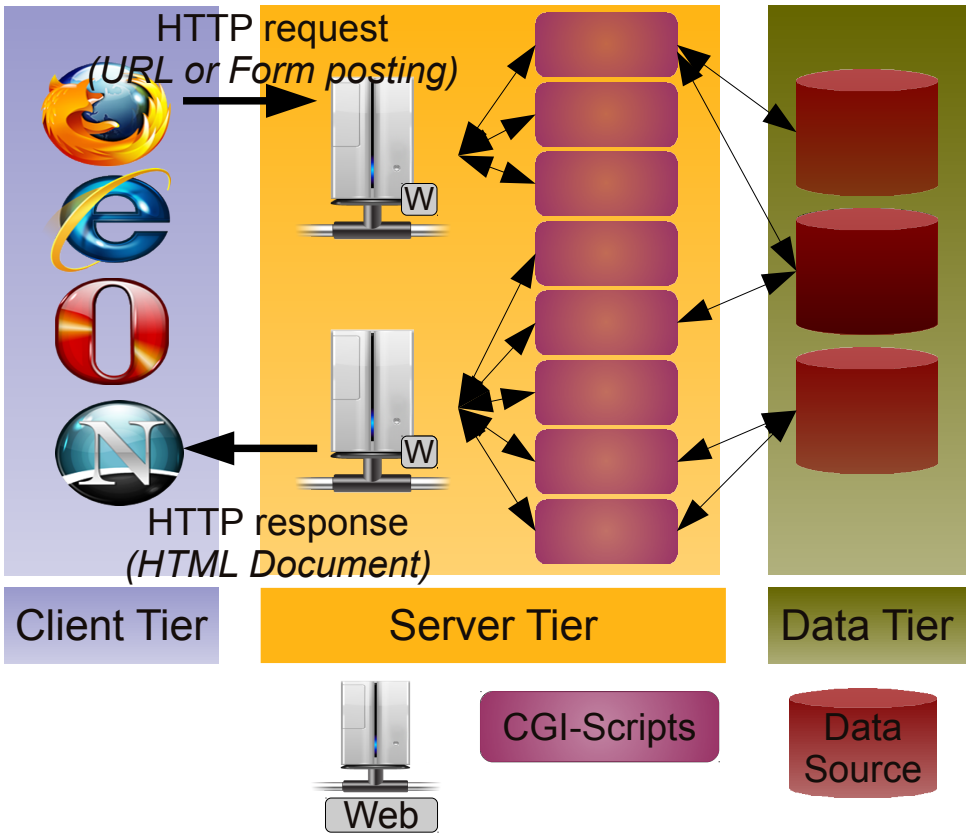
First Generation



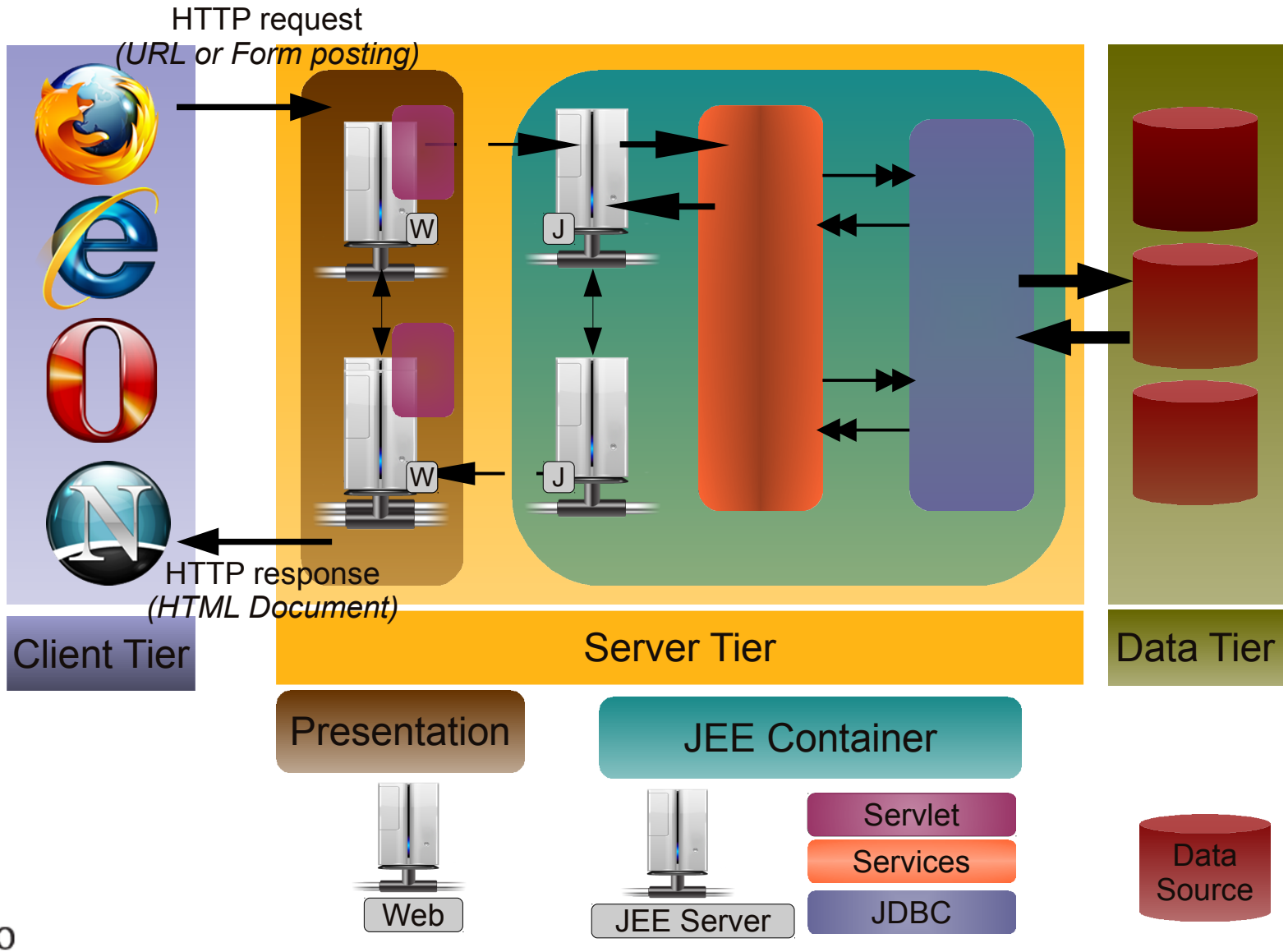
First Generation

Observations

- simple design
- client-tier
  - building blocks are?
- business tier
  - aggregation of scripts
  - scripts are
    - independent
    - ***stateless***
- lacks organic growth
- security nightmare



Second Generation



Second Generation

Observations

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