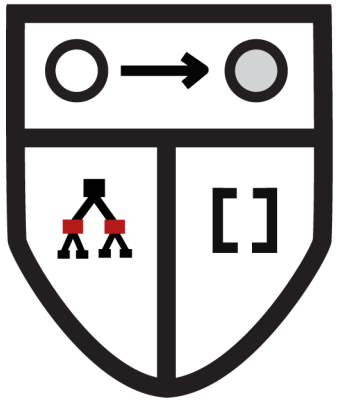


CS 3110

Software Architecture

Prof. Clarkson

Fall 2016

Today's music: *Thru' These Architects Eyes* by David Bowie

Review

Previously in 3110:

- language features for modularity: modules, structures, signatures, abstract types, includes, functors
- aspects of modularity: namespaces, abstraction, code reuse

Today:

- architecture and of large programs
- (until after Prelim 1: no new language features) 🎉🎉🎉

ARCHITECTURE

Architecture

- Any large system must be divided into sub-systems
- Goal of **architectural analysis**: identify sub-systems, their interfaces, how they interact
- Architecture is the highest-level design of software system

Elements of architecture

- Code **components**
  - a unit of an executing system
  - e.g. a server, a database
- Externally visible **properties** of those components
  - aspects of its functionality
  - e.g., services provided, data maintained, performance characteristics
- **Relationships** among components
  - e.g., implemented-by, shares-data-with, independent-of

Why analyze architecture?

- **Understanding**
  - Communicate system design between implementers, testers, maintainers, clients, users
  - Reduce system to a few parts; abstract from details; simplify
    - Working memory: humans can pay attention to only a small number of things at a time (3 or 4? 7?)
- **Reuse**
  - Identify what components can be repurposed from other systems
  - Assembly line model: cheaply produce system out of stock components
    - e.g., web mashups, 3110 website
- **Construction**
  - Division of (independent) labor
  - How to add new features

Ex: Architecture of web survey system

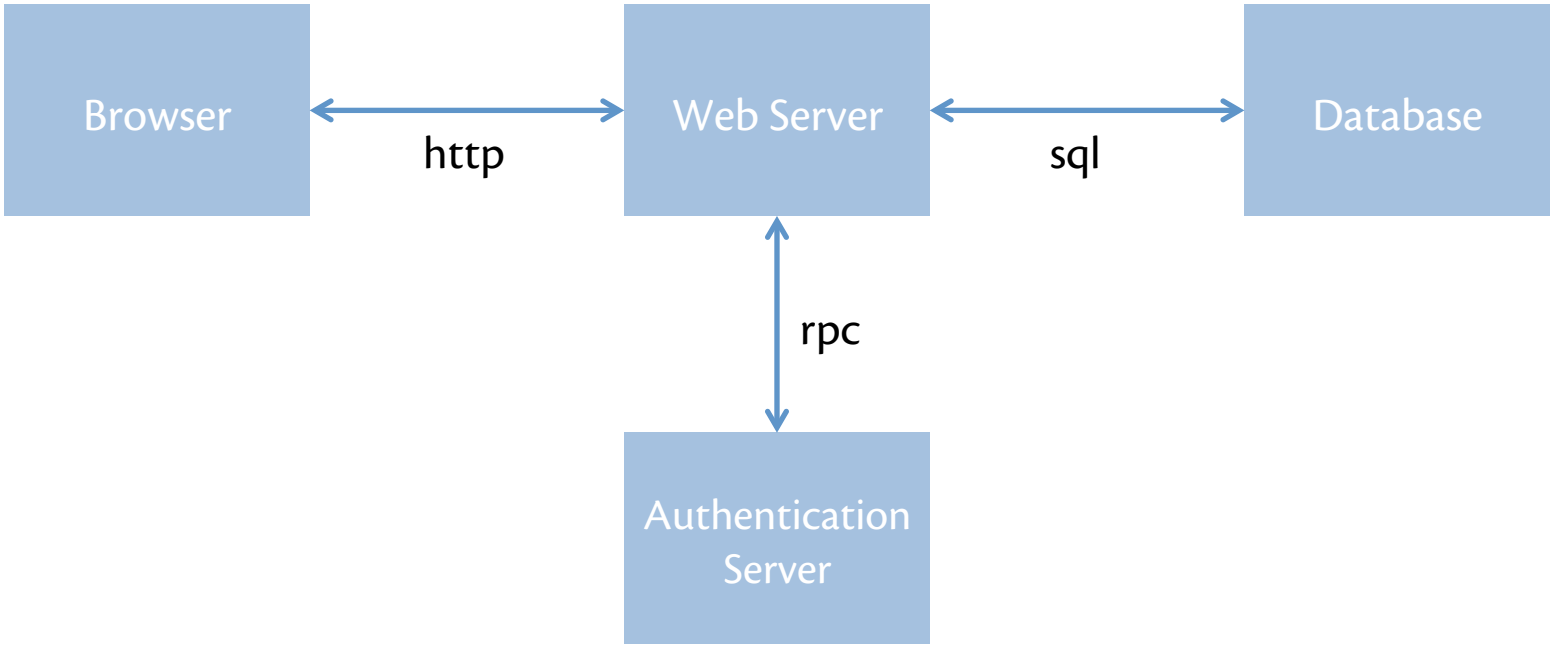
Requirements:

- present multiple-choice questions to user
- collect and store answers
- present results-in-progress to user after they submit

box and arrow diagram, aka
component and connector (C&C) diagram

Ex: Architecture of web survey system

New requirement: only some users are authorized to take survey; must authenticate users before they can register response



Ex: Architecture of web survey system

Examples of abstraction:

- No discussion of the code modules that make up components
- No details about the connectors (URLs, schema for SQL queries)

Examples of specification:

- Survey taker uses web browser
- Server must speak with all other components

Building blocks of architecture

Components:

- Computation elements or data stores
- Primarily from the view of **run time**: what happens while system is executing?
- Not necessarily from the view of **compile time**: how is code physically organized?

Building blocks of architecture

Connectors:

- Protocol: agreed upon means of communication
  - e.g., TCP, function call
- Topology could vary: binary, broadcast, ring, ...

Architectural patterns

- Architecture is a high-level creative activity, not a science
- Some common patterns:
  1. Pipe and filter
  2. Shared data
  3. Client–server

Ex 1: Pipe and filter architecture

- **Filter:** component that transforms data
  - receives data on input pipes
  - sends output data over pipes to other filters
  - might have >1 inputs, >1 outputs
  - each filter is independent of others and could operate concurrently
- **Pipe:** connector that relays data
  - unidirectional
  - does not change data
  - pipes handle storage, synchronization, rate of transfer, etc.

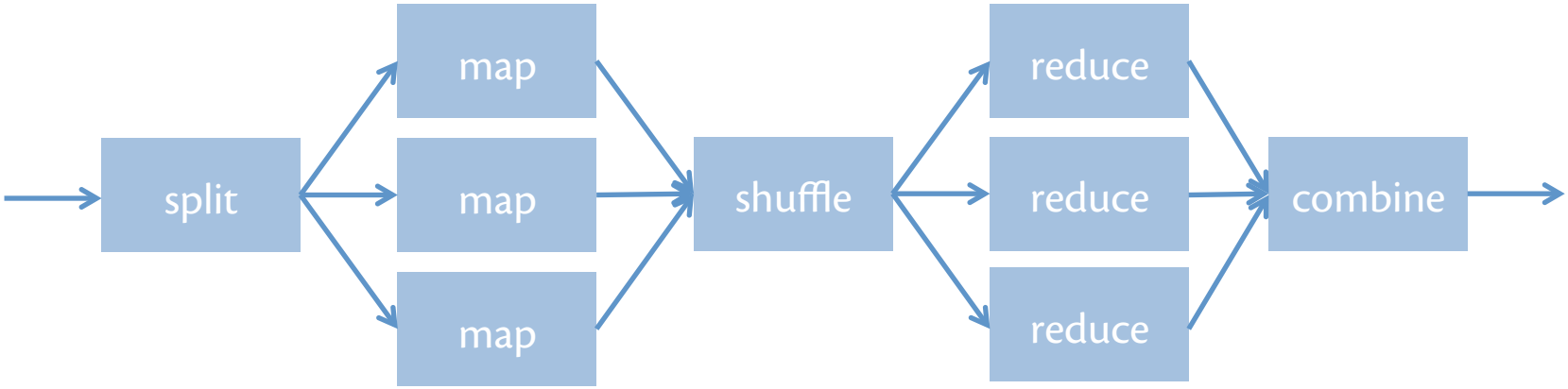
Ex 1: Pipe and filter architecture

MapReduce:

- Large amount of data comes in
  - e.g. documents whose words we want to index
- Split across multiple *workers* who concurrently process a block of data; output a *map* from keys to values
  - e.g. the key is a word, the value is the set of documents in which it appears
- Mapper outputs are *shuffled* to bring keys together at a worker
  - e.g., all the key-value pairs for a single word are brought together at a single worker
- Key-value pairs are *reduced* concurrently by workers; output new values
  - e.g., aggregate the sets of documents into a single set
- Values are *combined* into final output
  - e.g., the index: a map from all words to the set of documents in which they appear

Ex 1: Pipe and filter architecture

MapReduce as a pipe and filter architecture:



Ex 2: Shared data architecture

- **Data repository:** component that stores data
  - provides reliability, backup, access control
  - might be passive or might actively notify accessors about changes in data
- **Data accessor:** component that does computation with data
  - gets data from repository, computes, puts data back to repository
  - accessors do not directly communicate with one another
- **Interfaces:** connectors that gives read/write access to repository

Ex 2: Shared data architecture

PeopleSoft as a shared data architecture:

Ex 3: Client–server architecture

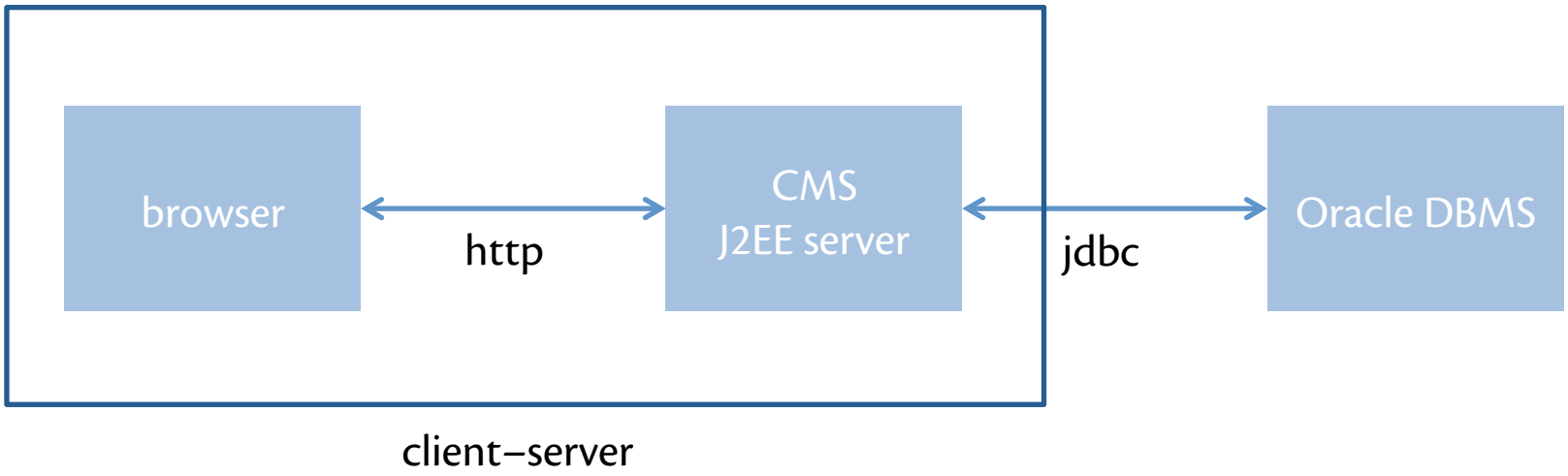
- **Server:** component that provides service/resources
  - When server provides a storage service, might reduce to shared data arch.
- **Client:** component that accesses service/resources
  - clients do not directly communicate with one another
  - clients need not be co-located with server
- **Channels:** connectors that allow client to make request, then server to return response
  - asymmetric: client can contact server, not vice-versa
  - (a)synchronous: client waits for response?
- Generalizes to *n-tier architecture*, in which server acts as client to another server, etc.

Ex 3: Client–server architecture

CMS as client-server architecture:

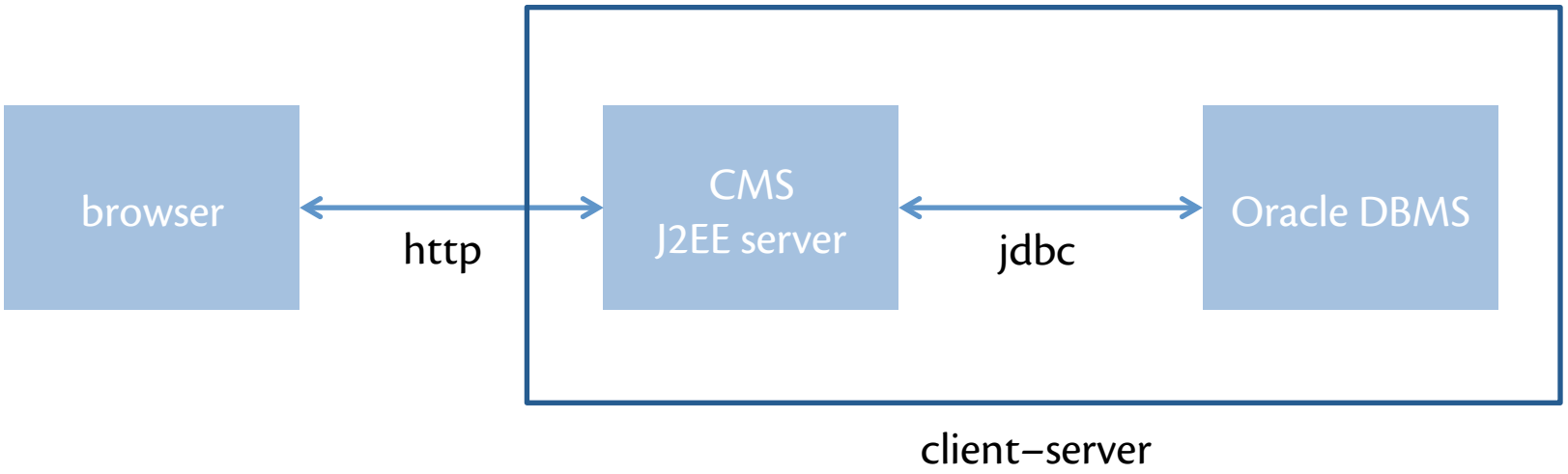
Ex 3: Client–server architecture

CMS as client-server architecture:



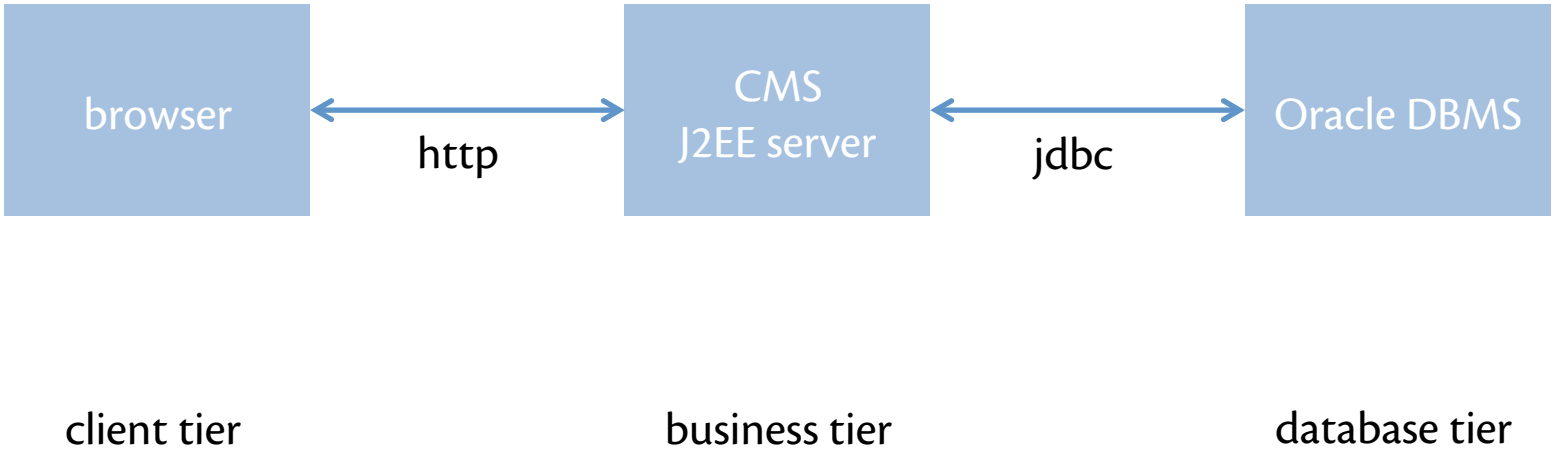
Ex 3: Client–server architecture

CMS as client-server architecture:



Ex 3: Client–server architecture

CMS as 3-tier architecture:



Question

Which architecture best describes the Enigma cipher?

A. Pipe and filter
B. Shared data
C. Client–server
D. None of the above
E. YNOXQ

Question

Which architecture best describes the Enigma cipher?

A. **Pipe and filter**
B. Shared data
C. Client–server
D. None of the above
E. YNOXQ

From architecture to design

- Architecture *is* a kind of design
  - focuses on highest level structure of system
  - based on principle of divide and conquer
- But architecture isn't about code per se
- As the *design process* iteratively proceeds, we get closer and closer to code
- *Design* as a phase of software development has a more specific connotation:
  - **System design:** decide what modules are needed, their specification, how they interact
  - **Detailed design:** decide how the modules themselves can be created such that they satisfy their specifications and can be implemented

Upcoming events

- [Wed] A2 due

This is architected.

THIS IS 3110

Acknowledgment

Parts of this lecture are based on this book:

Pankaj Jalote. *An Integrated Approach to Software Engineering*, third edition. Springer, 2005.