CSCI 3130 Software Architectures 1/3

February 5, 2013

Software Architecture

- What is a Software Architecture?
  - The description of the structure of a software system, which is composed of software elements, their externally visible properties and their relationships to each other.
  - Software system design at the highest level.
  - Closely related to Software Design – boundaries are very fuzzy.
  - Iterative and incremental
  - The is no one unique architecture for a given problem
- Why do we need it?

Understanding and Communication

- Software systems are too complex
- Abstraction of details
- Break a complex system into smaller, less complex sub-systems (Divide & Conquer)
- Individual sub-systems are better understood
- 50% of your time you deal with people who probably don't understand you.

"Buffalo buffalo Buffalo buffalo buffalo buffalo buffalo Buffalo buffalo."

Understanding and Communication

"[Those] (Buffalo buffalo) [whom] (Buffalo buffalo) buffalo, buffalo (Buffalo buffalo)."

Reuse

- Identifying the individual parts of the system facilitates encapsulation
- Encapsulation facilitates reuse
  - Many small problems have been solved before
  - Sub-systems are designed and implemented for a specific purpose / task with generalized interfaces
  - Generalized interfaces allow the reuse of the same sub-system in a different complex system
- Software Product Lines
- The more reuse, the less money it costs, the safer your job

Construction and Evolution

- Individual sub-systems and well defined interfaces allow:
  - Independent development of multiple sub-systems in parallel by different teams
  - Independent testing of multiple sub-systems with much less test cases
  - Replace one implementation of a sub-system with another implementation
  - Easily estimate the impact of a change
- You won't get it right the first time → software evolves

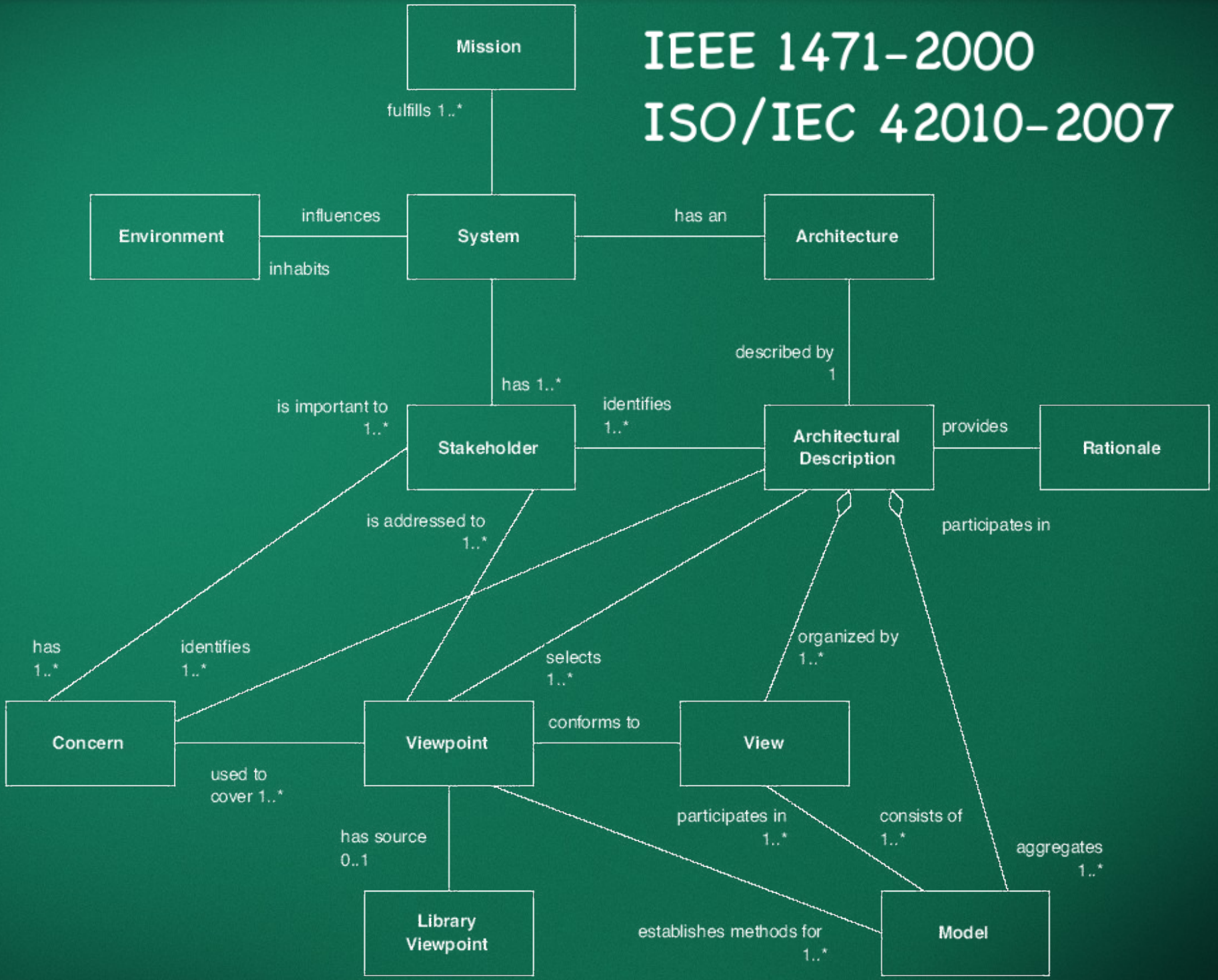
Analysis

- Up-front analysis prevents undesired surprises:
  - Design decisions
  - Performance requirements
  - Reliability
  - Usability
- Does each sub-system satisfy its specific requirements? Under which conditions?
  - Implementation restrictions
  - Hardware
  - Storage
  - Interconnect
  - Support

Software Architecture Models

- Various formal models / frameworks exist:
  - 4+1
  - RM-ODP
  - SOMF
  - IEEE 1471-2000 - ISO/IEC 42010-2007 (standards)
- Languages to describe the architecture:
  - Acme
  - Wright
  - UML
- Pick and choose
- All have in common: **Views**

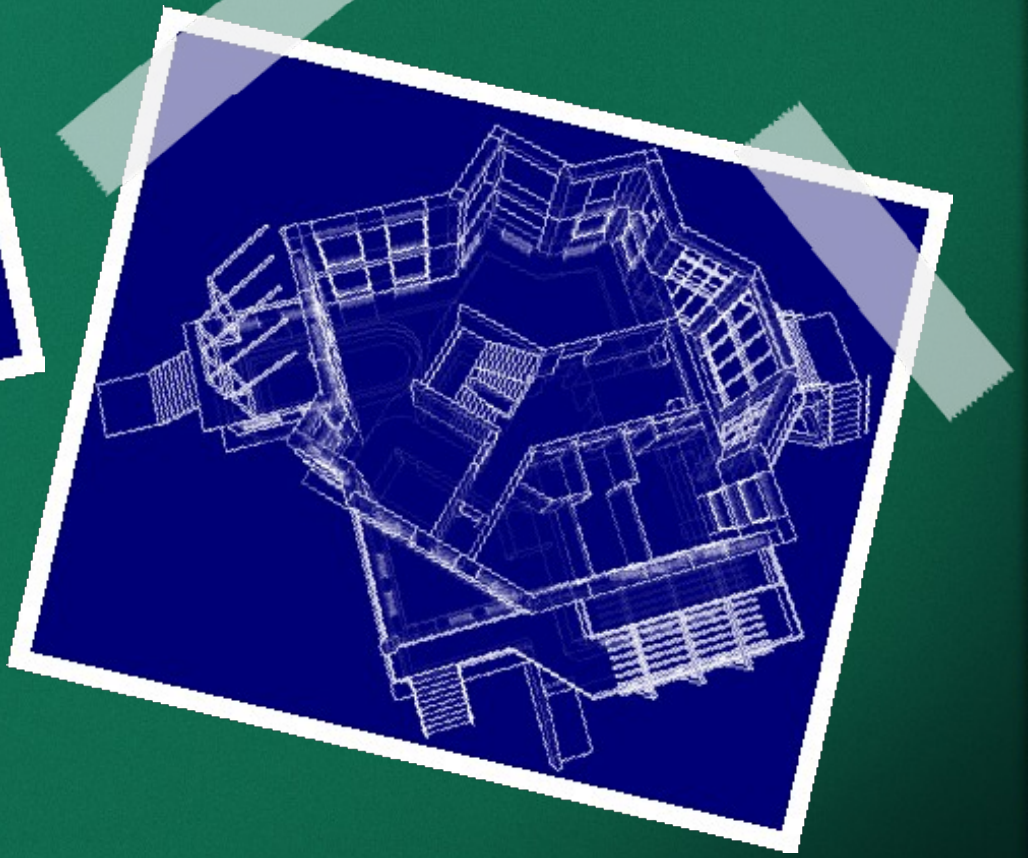
IEEE 1471-2000
ISO/IEC 42010-2007



Architecture Views

- Description of the architecture from different perspectives (viewpoints)
- Facilitates communication:
  - Business Owner
  - Client
  - Software Designer
  - Developer
  - System Builder
- Everyone has their own vocabulary

Architecture Views

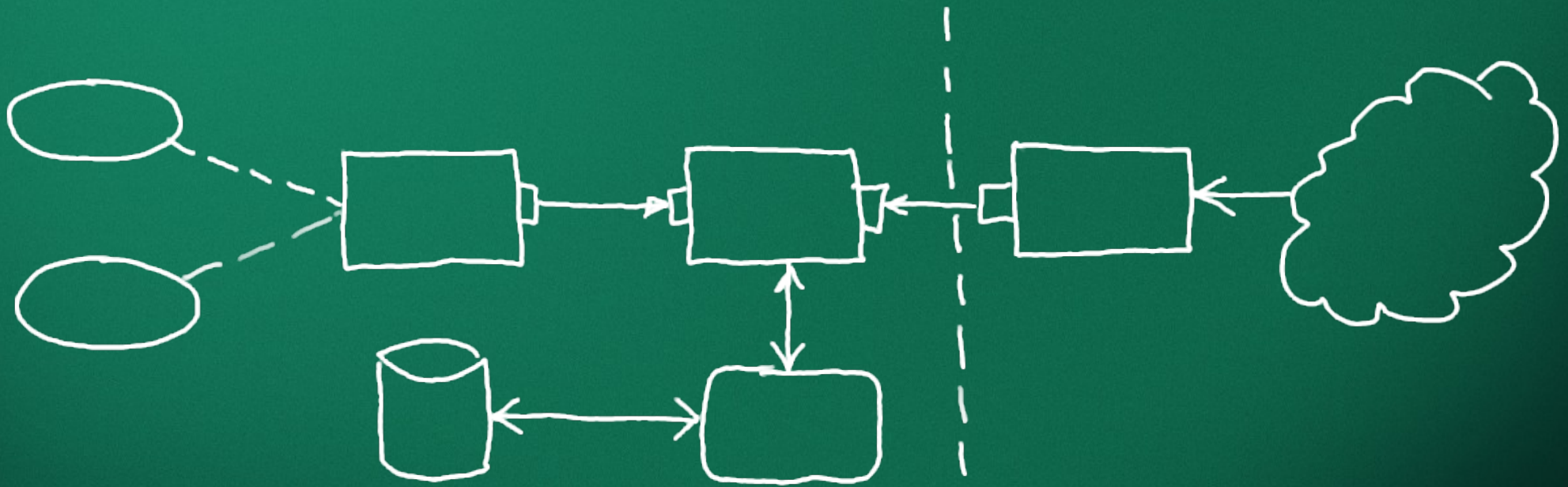


Architecture Views

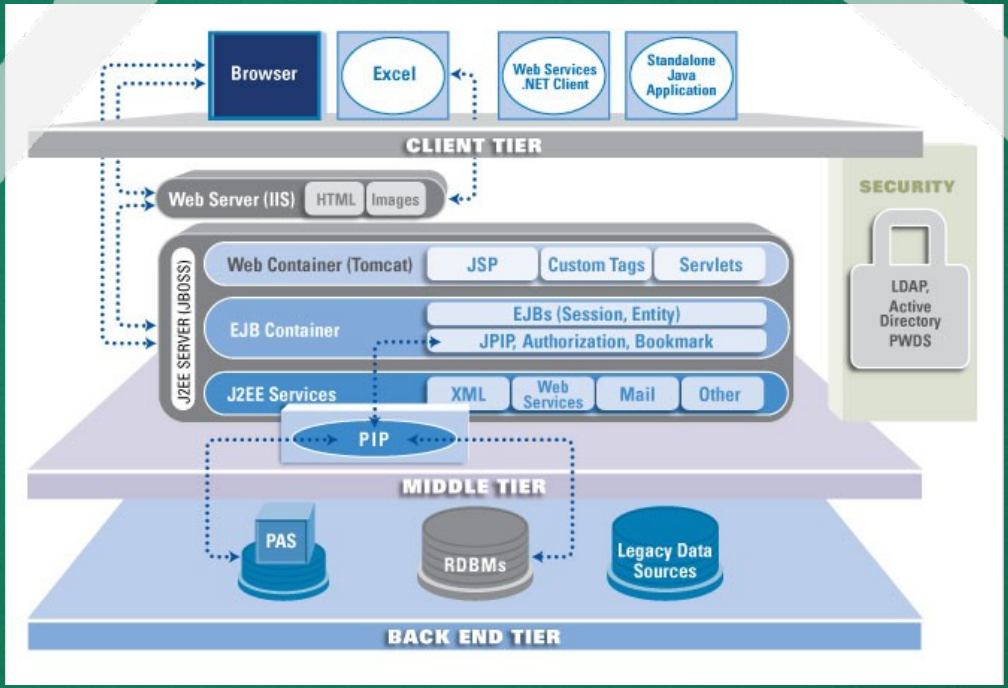
- Component & Connector View
  - Very universal, easy to understand, high-level
- Module View
  - Often the result of the software design
- Allocation
  - Used by integrators and system engineers

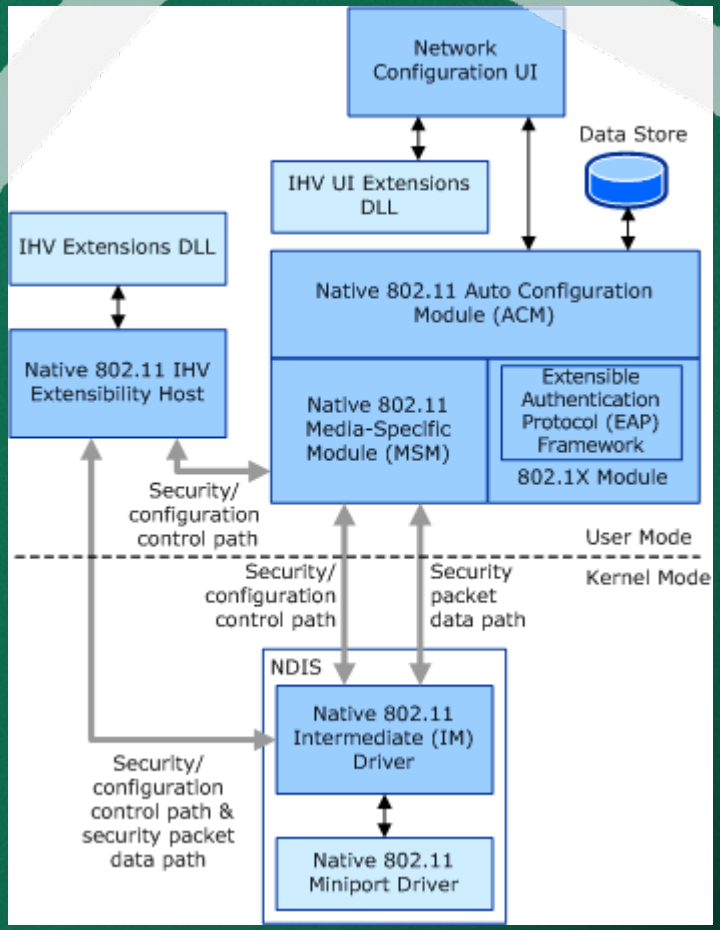
Component & Connector View

- Graph-like diagram of the parts of a system and their relationships
  - Parts = Components
  - Relationships = Connectors



Component & Connector View





Components

- Units of computation or data storage
- Distinct names – Choose them wisely!
- Components have types, the C&C view shows specific instances
- Interfaces (ports) to communicate with other components
- Describe components independent of the system

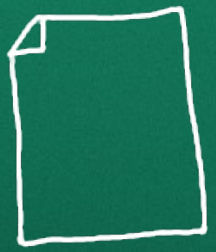
Component Types

Client

Server

Database / Repository

Application



Document / File

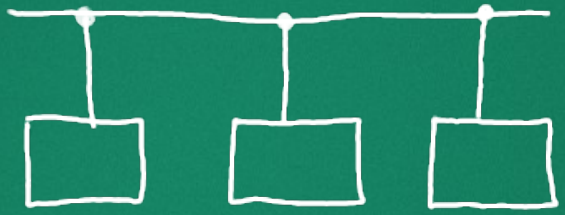
Connectors

- Connect components that interact with each other
- Distinct names – Choose them wisely!
- All communication between components is done through connectors – not only remote.
- Mechanisms:
  - Function call
  - RPC
  - Broker-based
- Provided and implemented by middleware
  - Beware: Easily slips into the component implementation!

Connectors

- Middleware connects components
  - Hardware (CPU instructions)
  - OS infrastructure (pipes, shared-memory)
  - Domain specific middleware (CORBA, HTTP, etc.)
- Different communication patterns and protocols
  - Point-to-point
  - Broadcast
  - Multicast
  - HTTP / REST
  - CORBA (IIOP), SOAP
  - AMQP
- Use different notation for different types of connectors

Connectors



Communication Bus

Request - Response

Pipe

Remote Procedure Call

Database Access

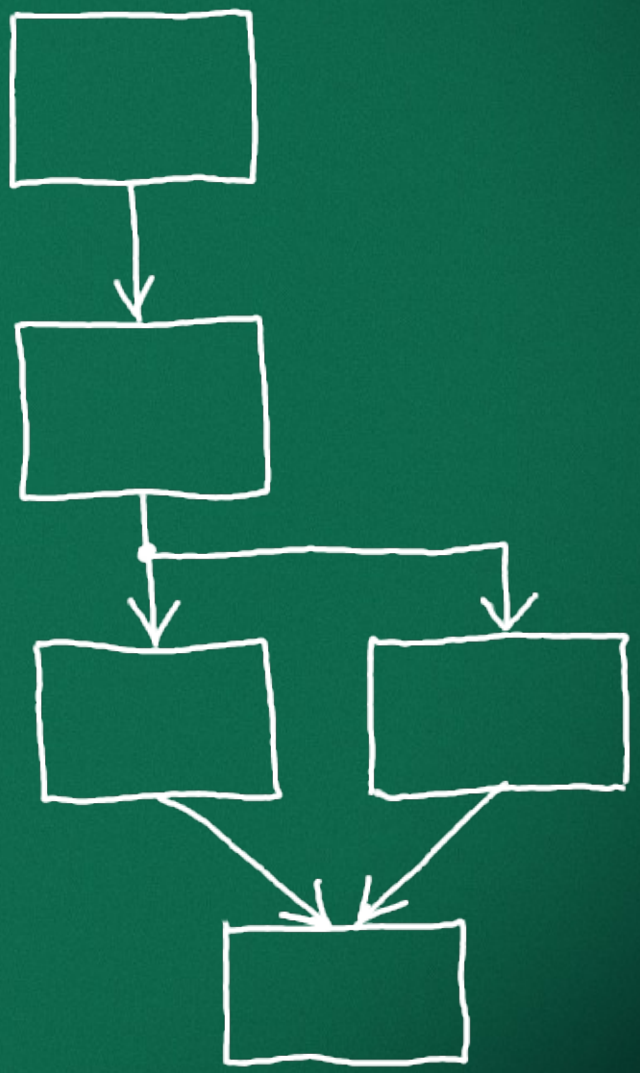
Example: Todo List

Architecture Styles

- Design Patterns for Software Architectures
- Best practices to solve common problems
- Architecture is a combination of many
- Module View (Software Design):
  - Decomposition
  - Uses
  - Generalization
  - Layered

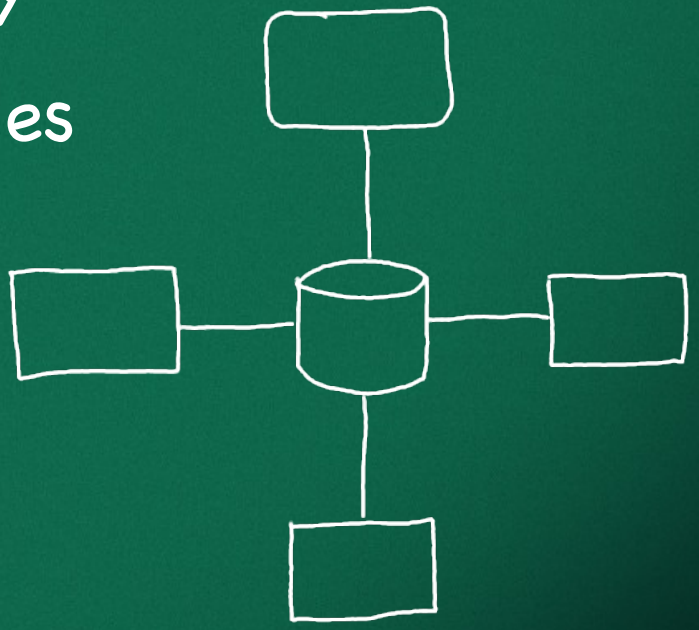
Pipe & Filter

- Producer-consumer pattern
- Good encapsulation
- Asynchronous processing at each component
- Pipe connector responsible for synchronization
- Parallel processing (Map/Reduce)
- Document processing, signal processing, ETL



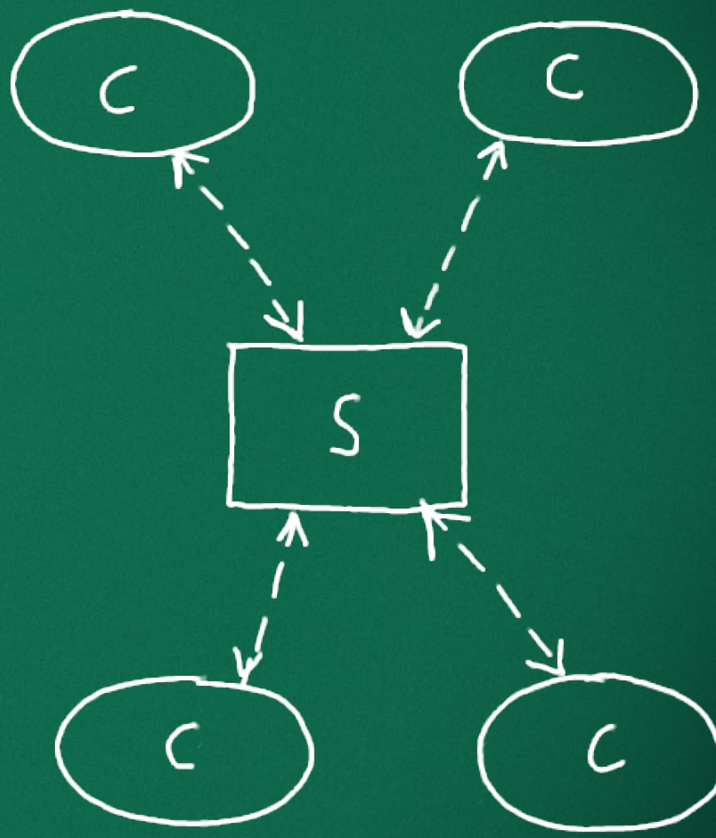
Shared-data

- Data repository + data accessors
- Communication through data repository
- Data repository responsible for data consistency and synchronization
- Add / remove components easily
- Passive / active data repositories
- Database applications, Web applications



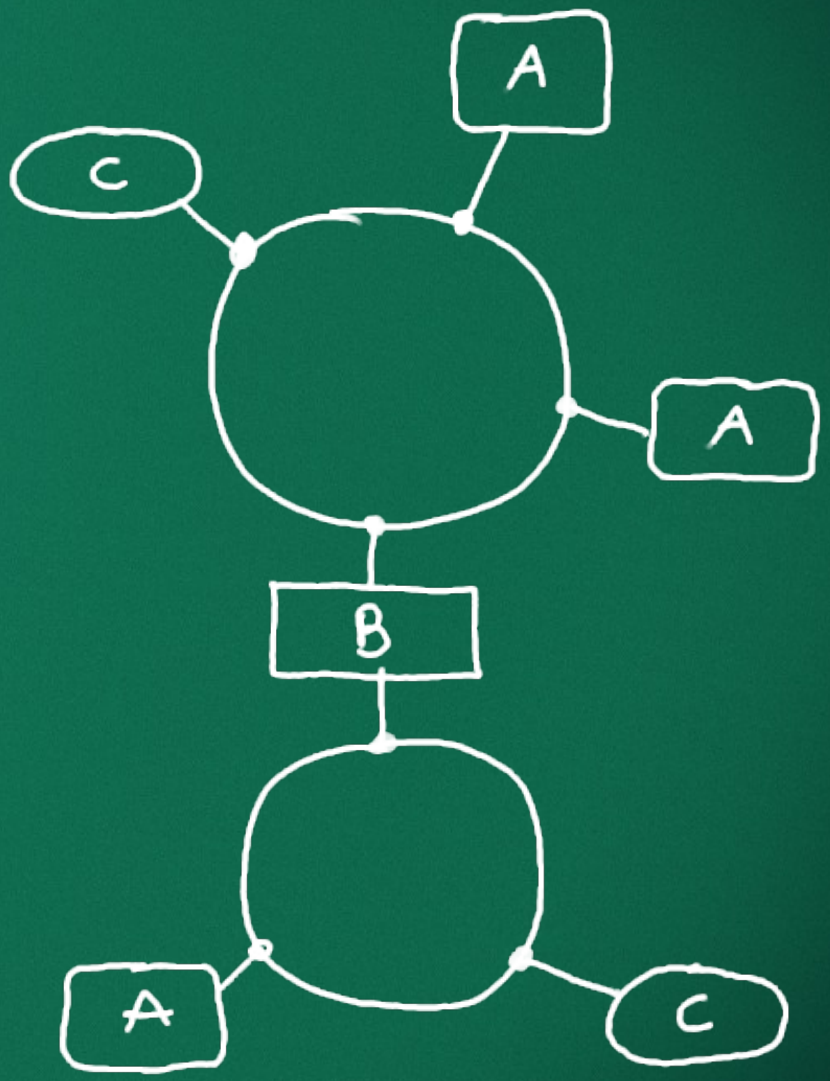
Client-Server

- Client requests a response
- Response is generated by an action executed by the server
- Client waits for response
- Server itself might be a client
- Often stateless
- Client initiated
- Lightweight clients
- WWW, HTTP, REST



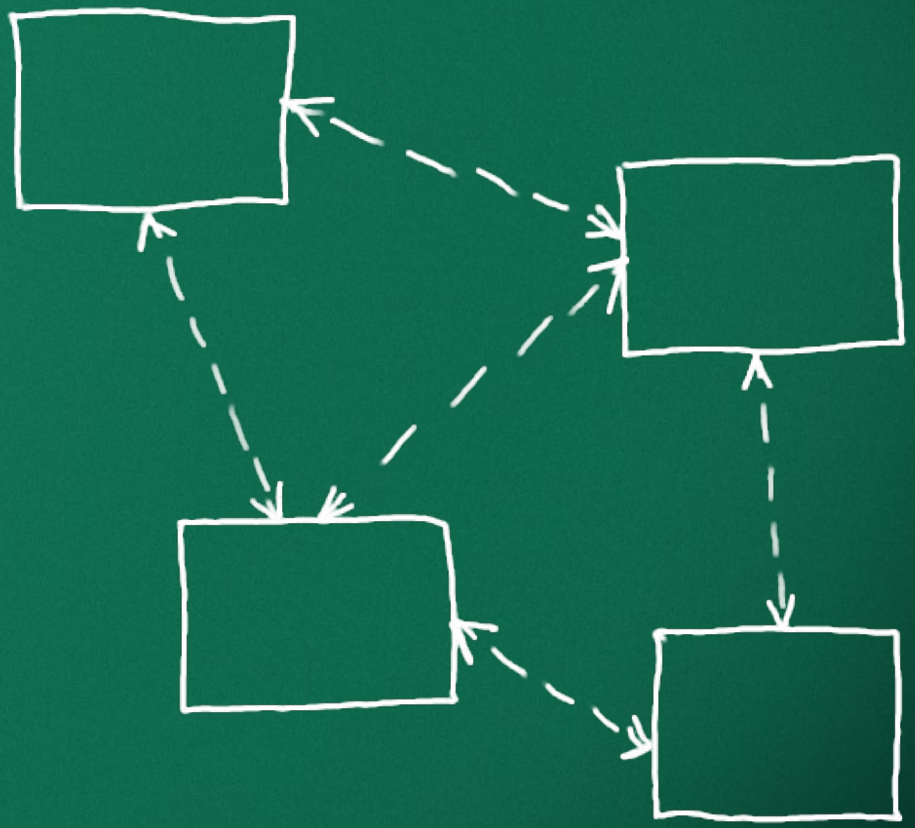
Publish-Subscribe

- Producers publish messages on a shared medium (e.g. message bus)
- Consumers subscribe to certain types of messages
- Brokers may connect independent bus systems
- Scalable, transaction safe, easily extensible
- IRC, ESB, AMQP



Peer-to-Peer

- Like client-server, but every component is both client and server
- Intermediate components can act as proxies and/or caches
- Distribution of load
- Highly scalable for specific applications
- ICP, CDNs, BitTorrent, Gnutella, etc.



EOF

- Tuesday, February 12, 2013:
  - Architecture Integrity: Why you should listen to the architect?
  - Architecture Analysis: What can you learn from an architecture?
  - Architecture Documentation: How to communicate an architecture?