Software Design

Software Engineering 2

(3103313-1)

Amirkabir University of Technology

Fall 1399-1400

Moving to Design

- Design activities focus on how to build the system; how the functionality specified in the analysis model will be implemented.
  - Major activity is to evolve the analysis models into a design
  - Goal is to create a blueprint for the design that makes sense to implement
  - Determine how and where data will be stored
  - Determine how the user will interface with the system (user interface, inputs and outputs)
  - Decide on the physical architecture

Software Architecture

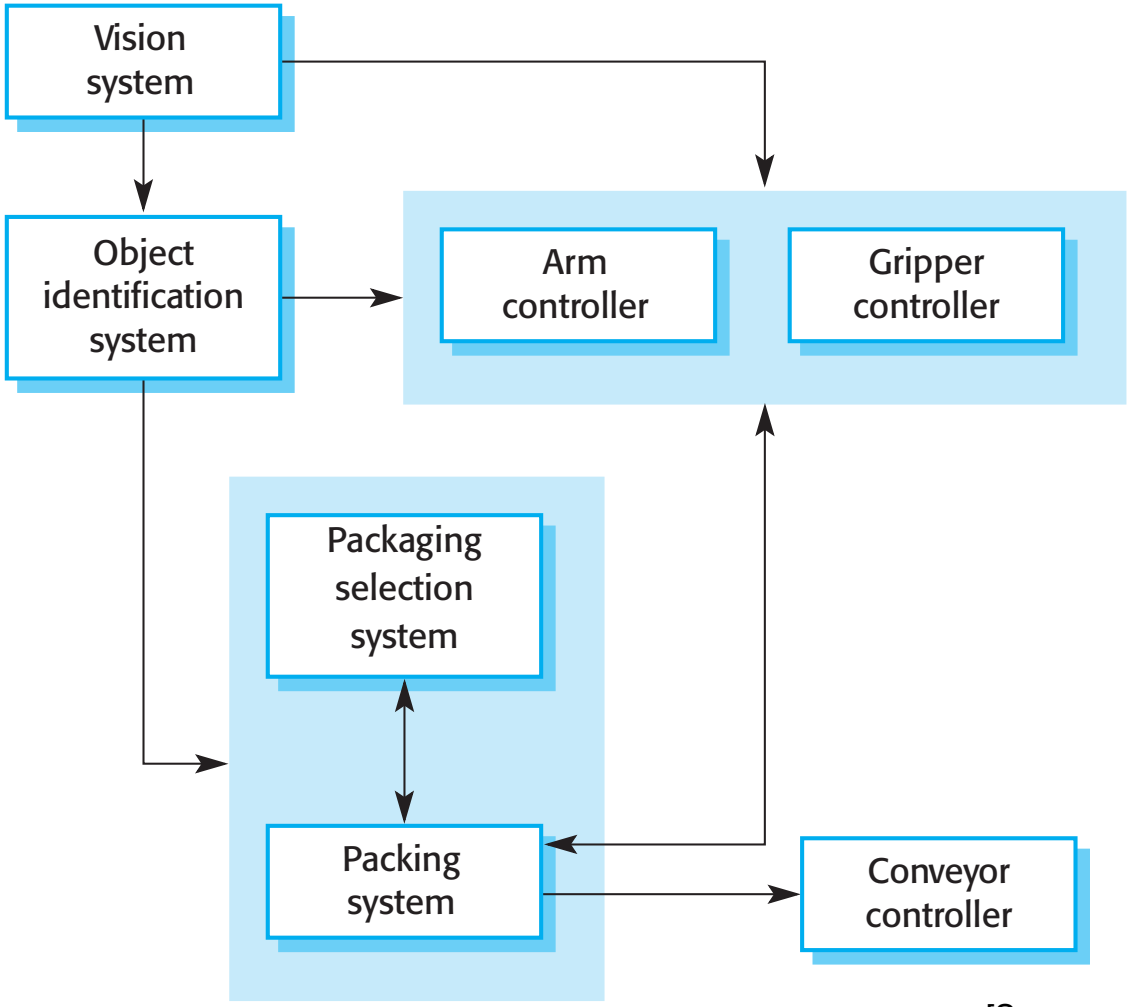
The IEEE definition of software architecture

"Architecture is the fundamental organization of a software system embodied in its components, their relationships to each other and to the environment, and the principles guiding its design and evolution."

--IEEE 1471

- Software architecture is about making fundamental structural choices which are costly to change once implemented.
  - overall organization, how the software is decomposed into components, the server organization, and technologies

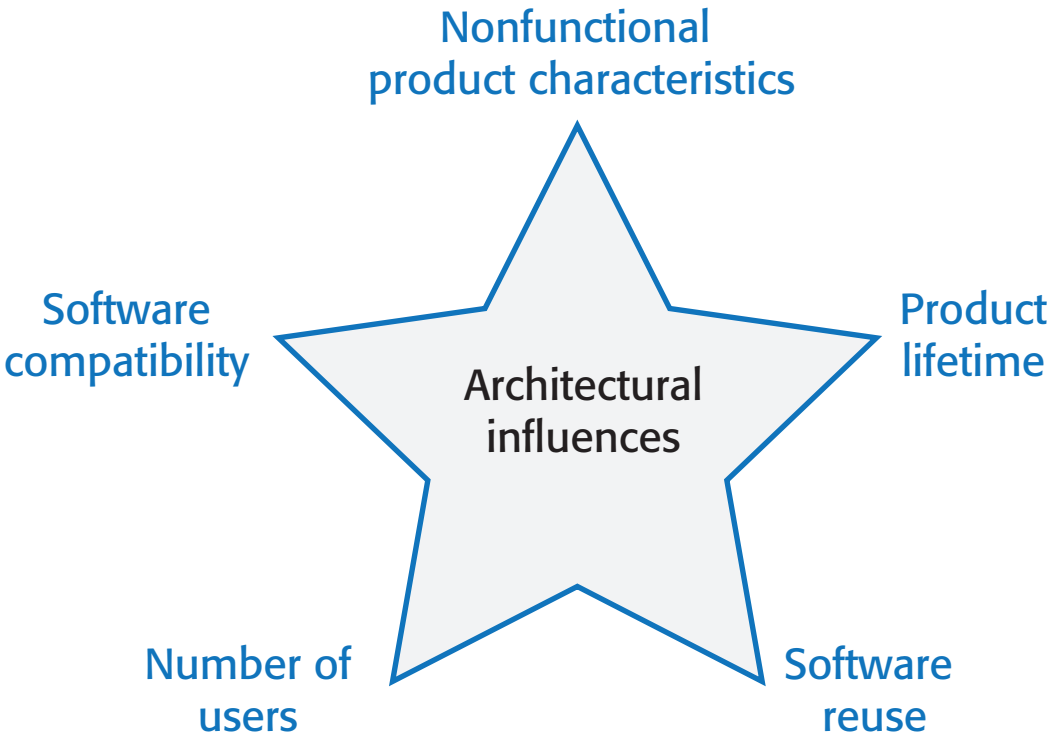
Software Architecture



[Sommerville, 2016]

Why Architecture?

Architectural design decisions

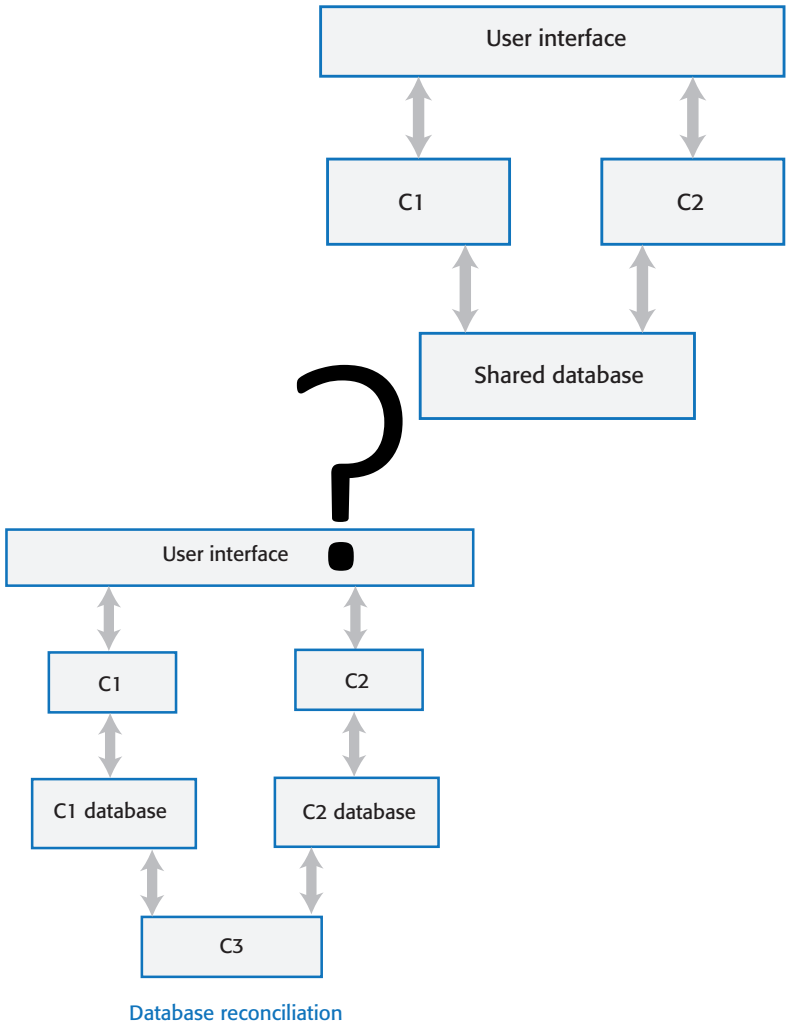


[Sommerville, 2020]

Trade-offs

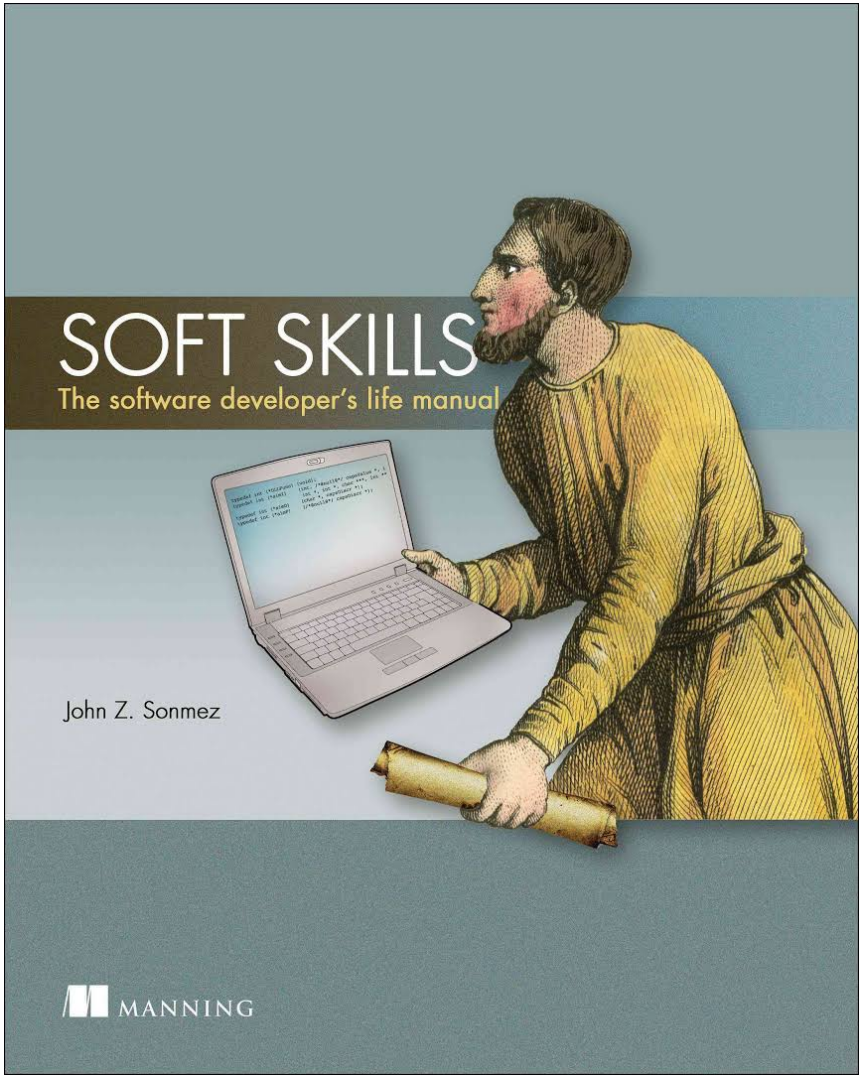
- Good-enough
- On time and budget

- maintainability vs. performance;
- security vs. usability;
- availability vs. time to market and cost.



[Sommerville, 2020]

break



Soft Skills
by John Sonmez

- For most software developers, coding is the fun part. The hard bits are dealing with clients, peers, and managers, staying productive, achieving financial security, keeping yourself in shape, and finding true love. This book is here to help.

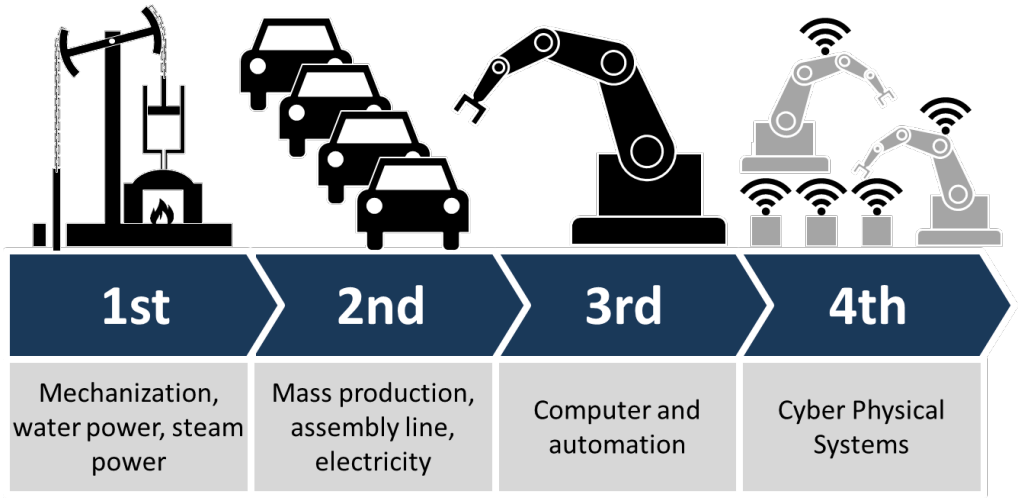
- The software developer's life manual is a guide to a well-rounded, satisfying life as a technology professional. Sonmez offers advice to developers on important **"soft"** subjects like career and productivity, personal finance and investing, and even fitness and relationships.

- Arranged as a collection of 71 short chapters.

* Forewords by Robert C. Martin (Uncle Bob) and Scott Hanselman.

Industry 4.0

- Industry 4.0 is a name given to the current trend of automation and data exchange in manufacturing technologies. It includes cyber-physical systems, the Internet of things, cloud computing and cognitive computing.



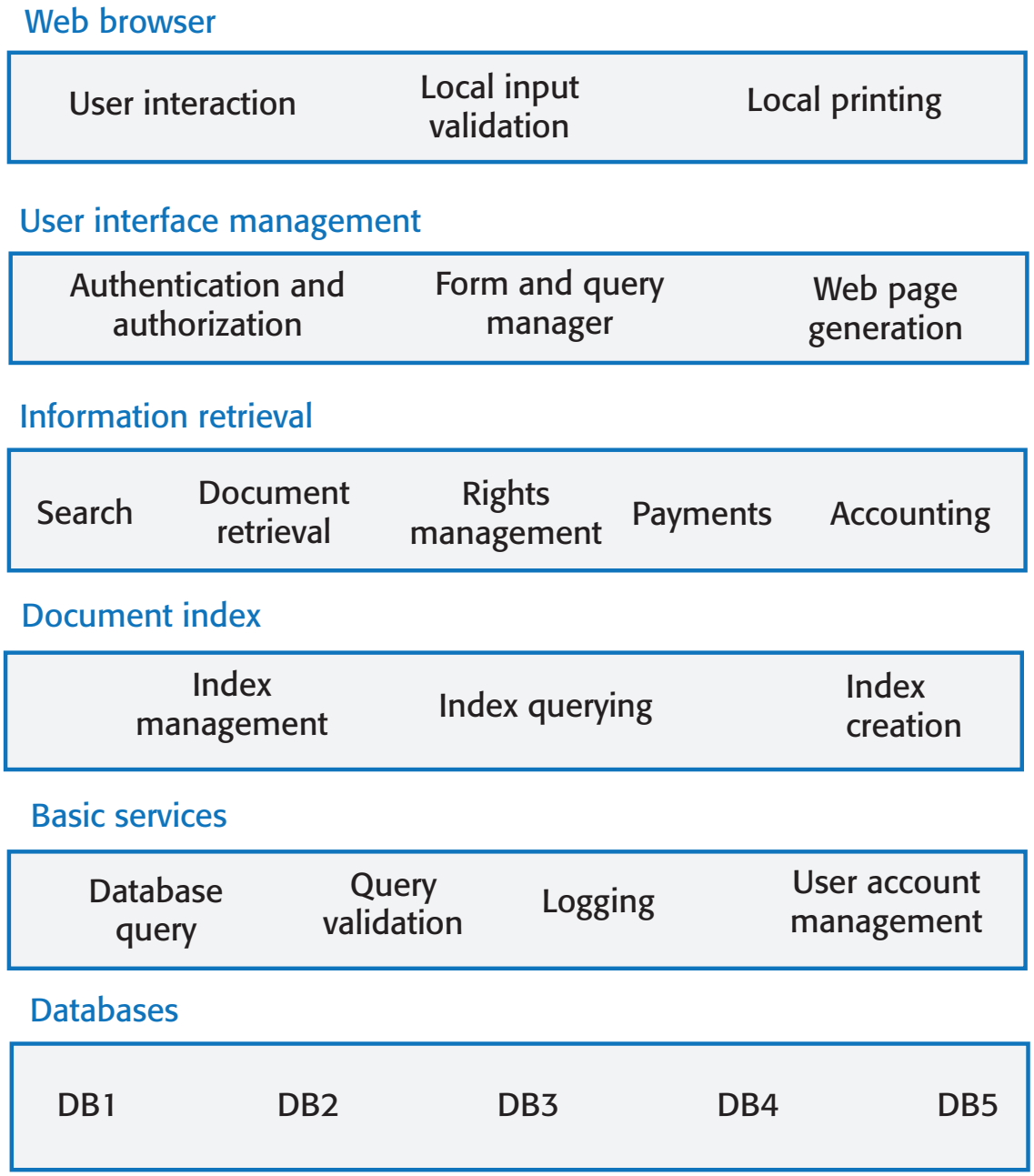
How Design the Architecture?

Similar Applications

Architectural Patterns/Styles

Influential Factors: Functionalities, Qualities, ….

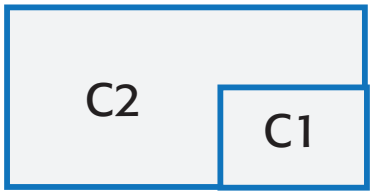
An architectural model of a document retrieval system



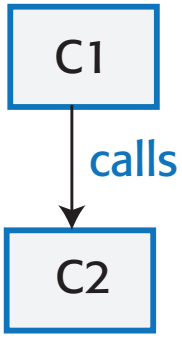
[Sommerville, 2020]

Architectural Complexity

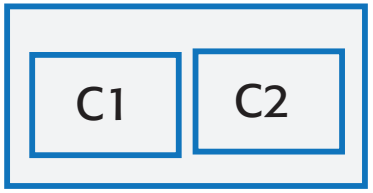
C1 is-part-of C2



C1 uses C2



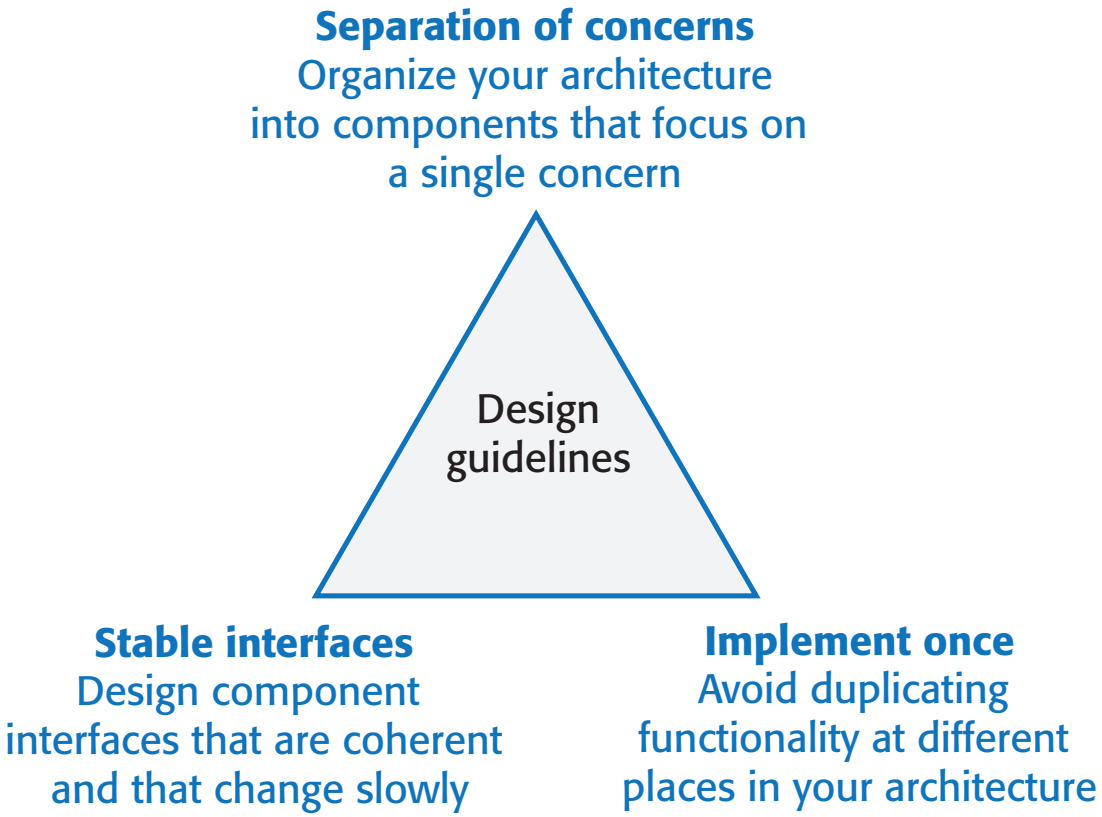
C1 is-located-with C2



C1 shares-data-with C2

[Sommerville, 2020]

Architectural design guidelines



[Sommerville, 2020]

Architectural Styles and Patterns

Common Architectures!

A stylized, abstract description of good practice, which has been tried and tested.

.

Architectural Styles and Patterns

Most Common Ones?

Architectural Styles and Patterns

Taxonomy

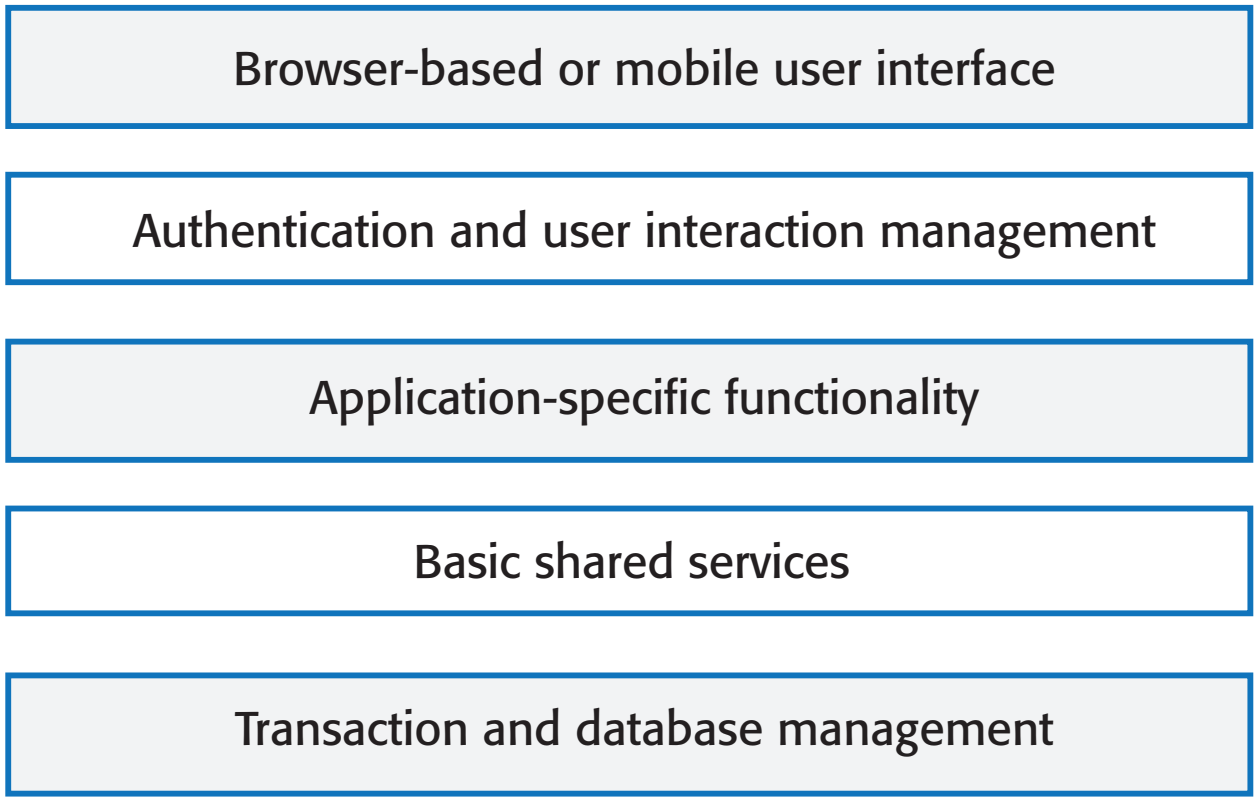
- Data-Centric/Repository
  - Blackboard
- Client-server (2-tier, 3-tier, n-tier, cloud computing exhibit this style)
- Component-based
- Layered (or multilayered architecture)
- Peer-to-peer (P2P)
- Pipes and filters
- Even-driven
- Service-Oriented
- Microservices
- MVC
- Asynchronous Messaging
- Distributed
- Call and Return
- Object-Oriented
- Monolithic
- Rule-based
- Mix and Match!

Web Applications

The most common architecture for web applications?

Web Applications

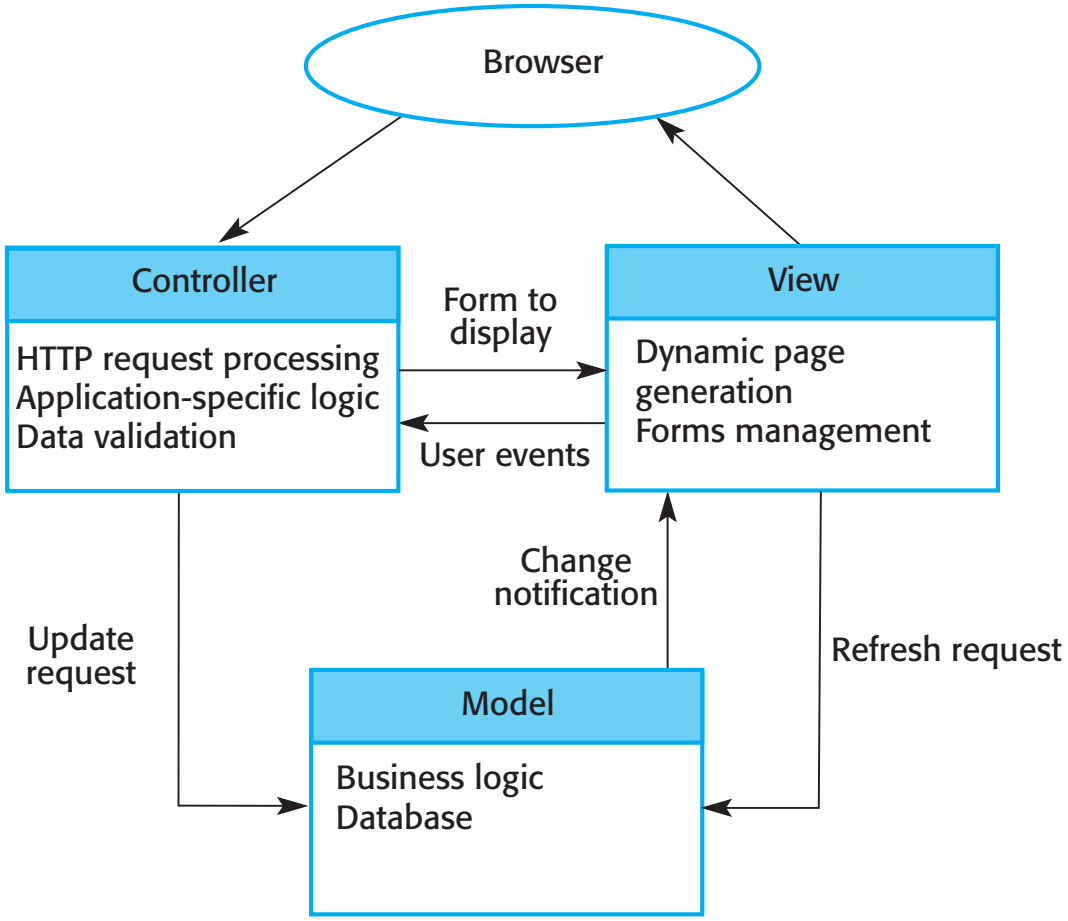
A generic layered architecture



[Sommerville, 2020]

Web Applications

MVC Pattern



[Sommerville, 2020]

Web Applications

X

... and

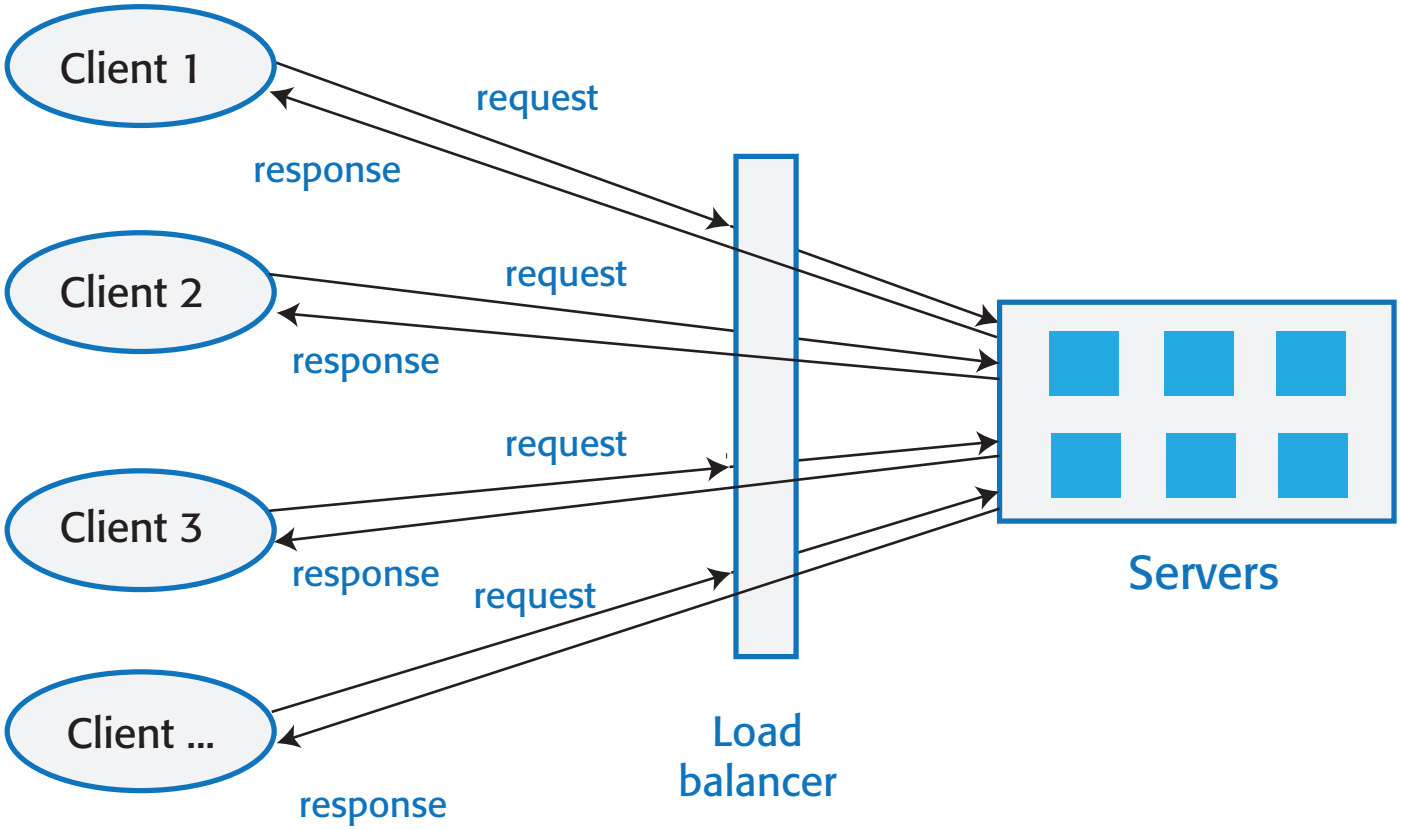
Architectural Styles and Patterns

Taxonomy

- Data-Centric/Repository
  - Blackboard
- Client-server (2-tier, 3-tier, n-tier, cloud computing exhibit this style)
- Component-based
- Layered (or multilayered architecture)
- Peer-to-peer (P2P)
- Pipes and filters
- Even-driven
- Service-Oriented
- Microservices
- MVC
- Asynchronous Messaging
- Distributed
- Call and Return
- Object-Oriented
- Monolithic
- Rule-based
- Mix and Match!

Distributed Architecture

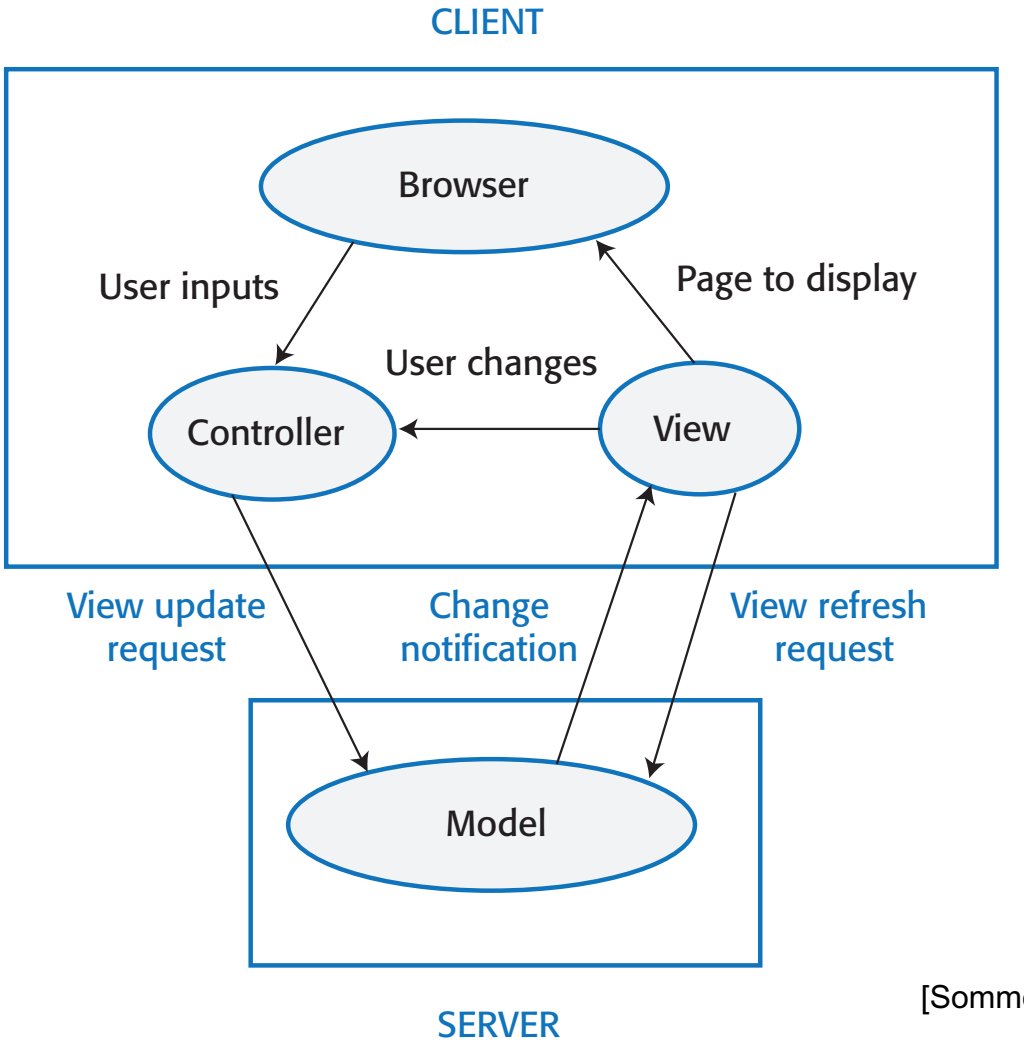
client-server



[Sommerville, 2020]

Distributed Architecture

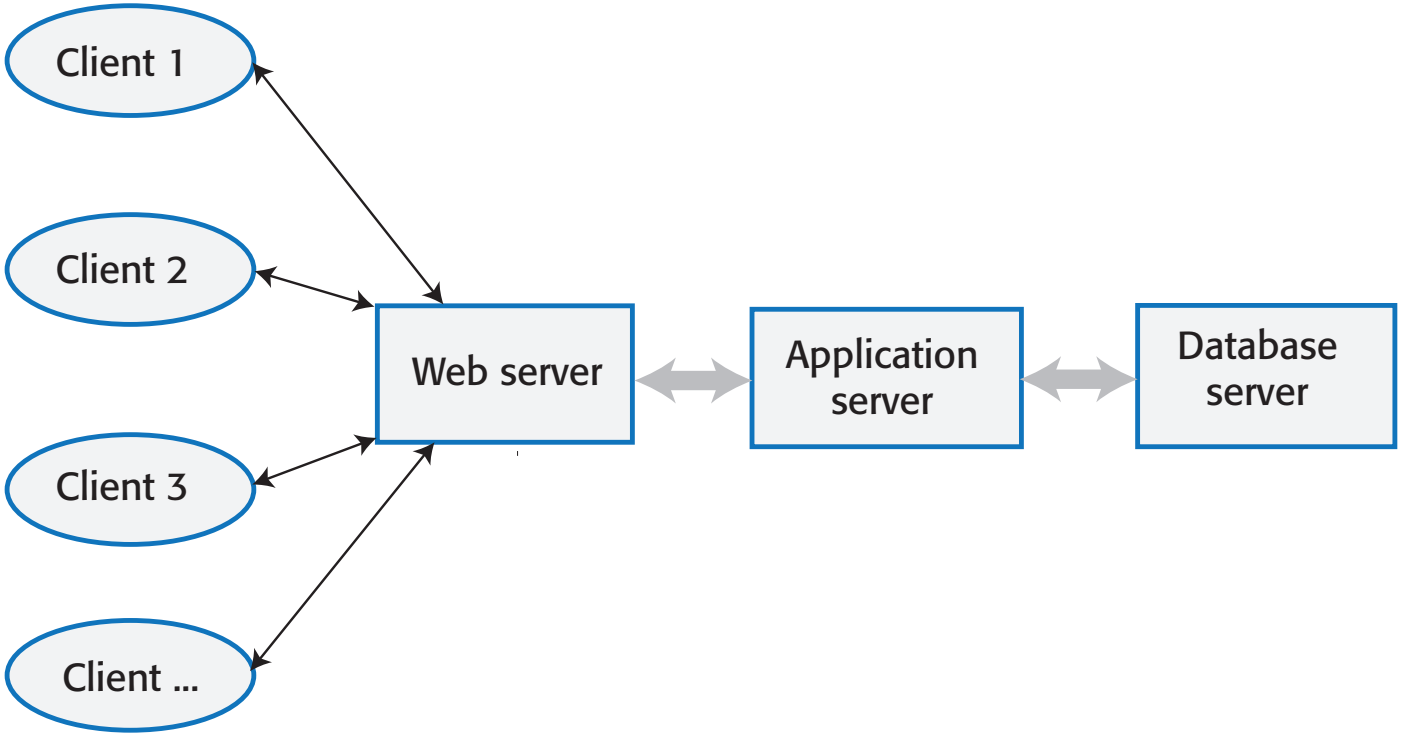
Model-view-controller pattern



[Sommerville, 2020]

Distributed Architecture

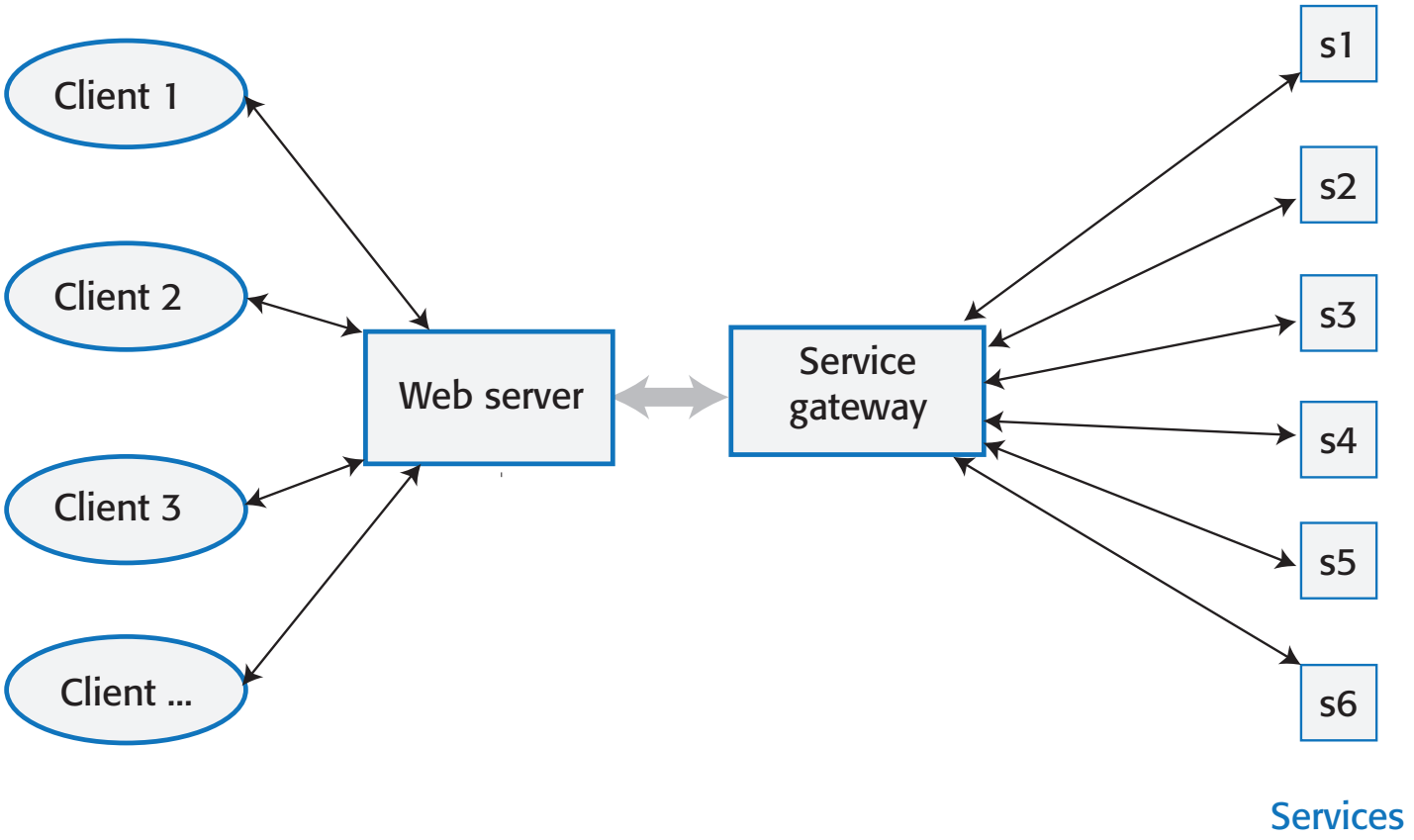
Multi-tier client-server architecture



[Sommerville, 2020]

Distributed Architecture

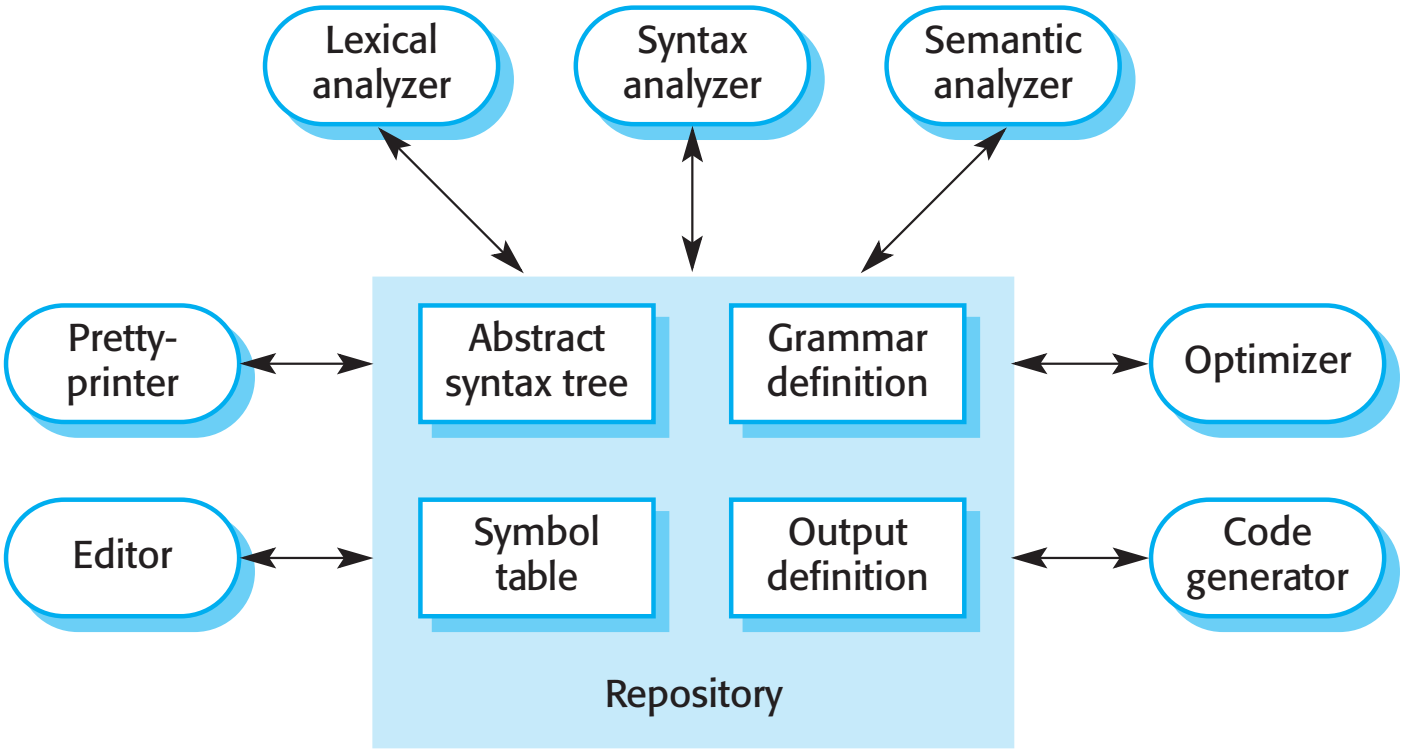
Service-Oriented Architecture



[Sommerville, 2020]

Data-Centric Architecture

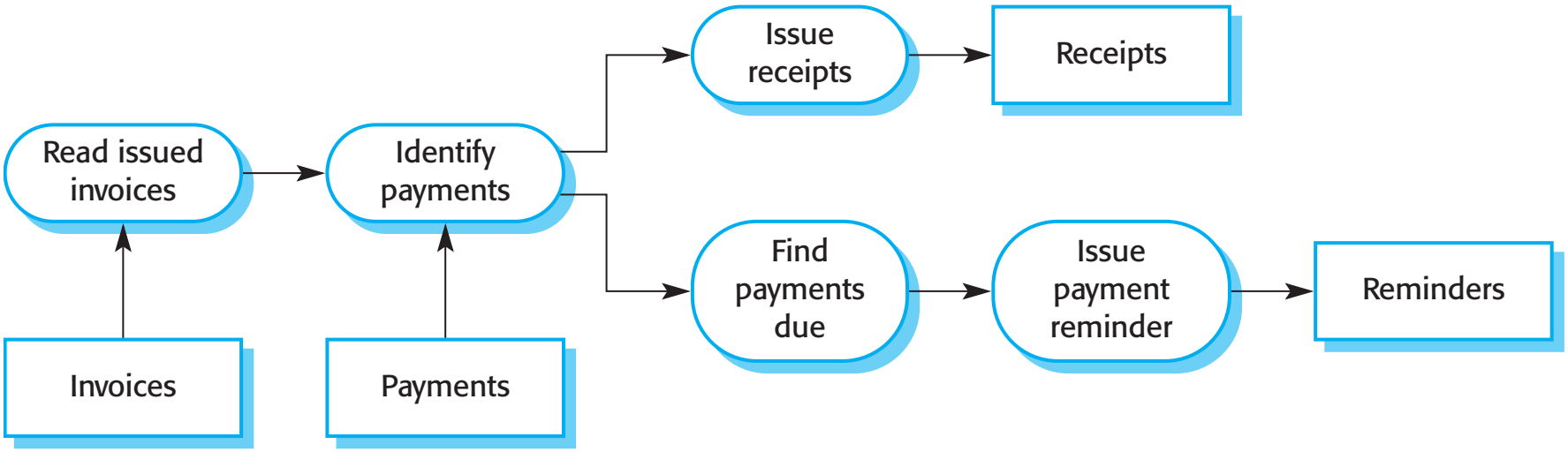
Repository Architecture



[Sommerville, 2016]

Pipes and Filter

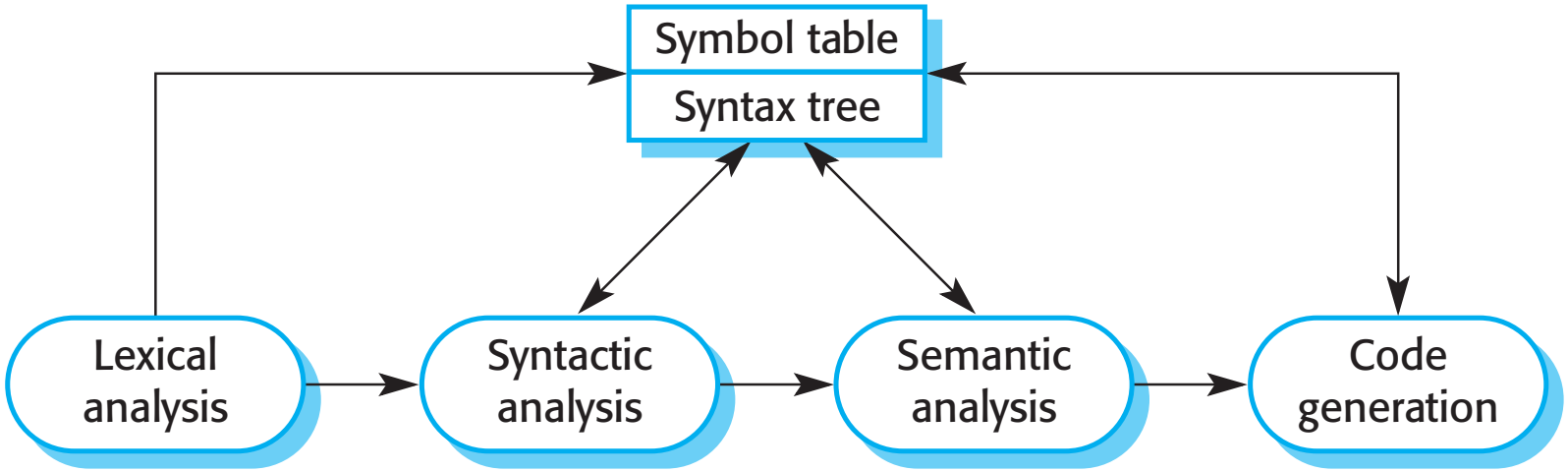
Data-processing systems



[Sommerville, 2016]

Pipes and Filter

Compliers' Reference Architecture



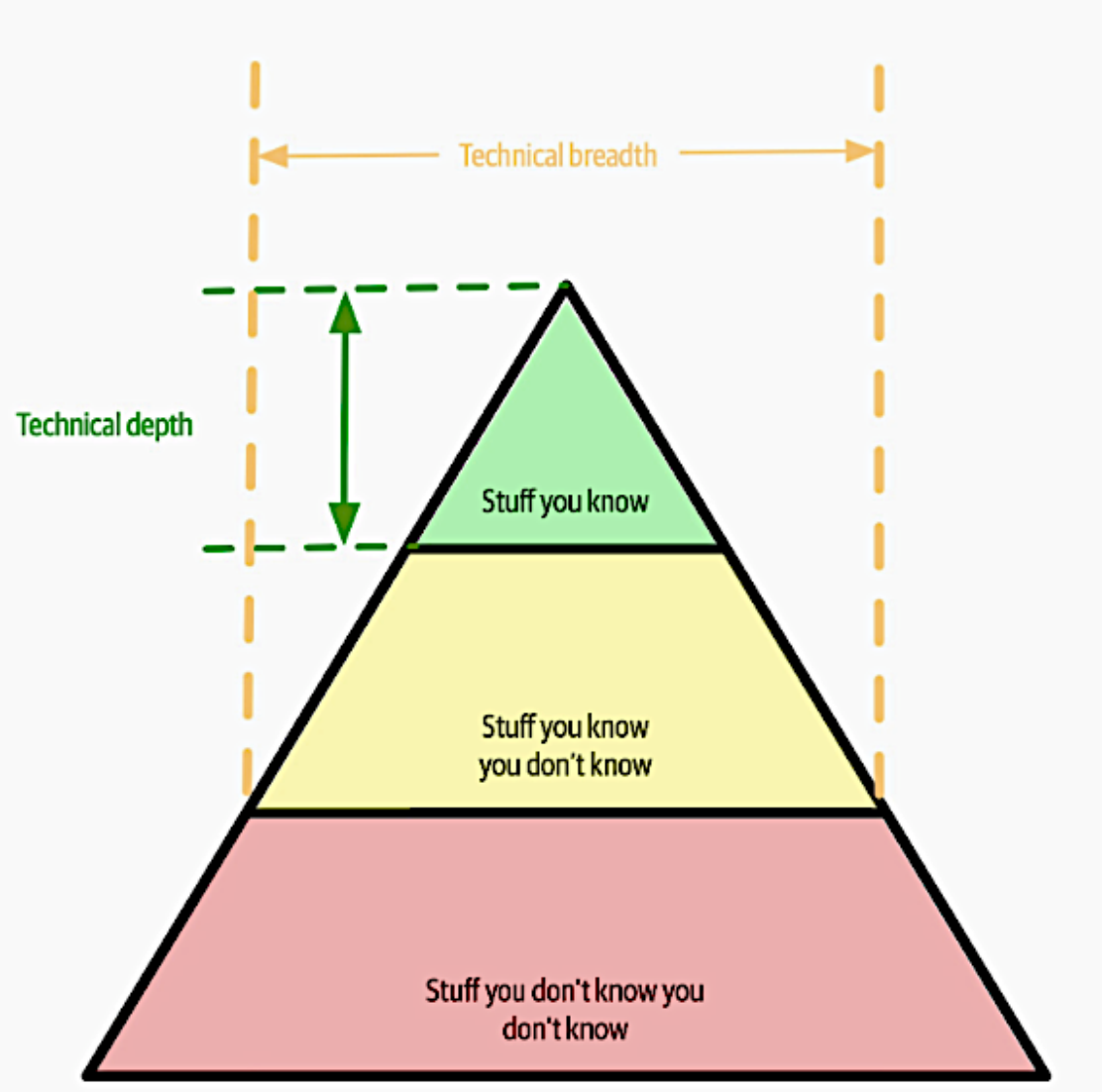
[Sommerville, 2016]

Read More

- Mark Richards, “Fundamentals of Software Architecture.”, O’reilly, 2020.

- I. Sommerville, Engineering Software Products: An Introduction to Modern Software Engineering, Pearson, 2019
- R. Stephens, Beginning Software Engineering, Wrox, 2015

break



The pyramid representing all knowledge”

Mark Richards. “Fundamentals of Software Architecture.”

GitLab

GitLab is a web-based DevOps lifecycle tool that provides a Git-repository manager providing wiki, issue-tracking and CI/CD pipeline features, using an open-source license, developed by GitLab Inc.

The **GitLab team handbook** is the central repository for how the company runs the company. Printed, it consists of over 5,000 pages of text. As part of the value of being transparent the handbook is open to the world, and they welcome feedback.

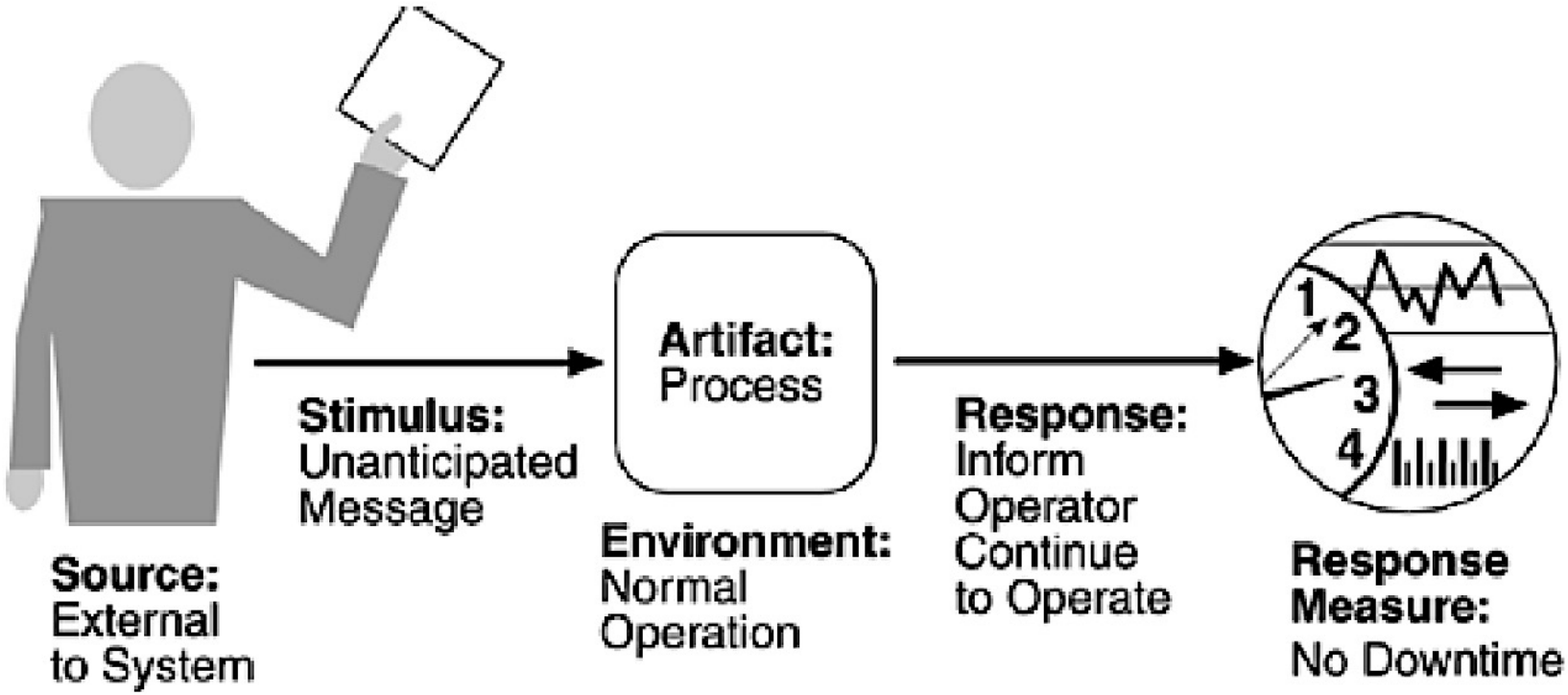
https://about.gitlab.com/handbook/

Measuring Architecture Characteristics
&
Achieving Qualities

- Objective definitions for architecture characteristics
- Measurable Features
  - Operational, Structural, Process

System Quality Attributes

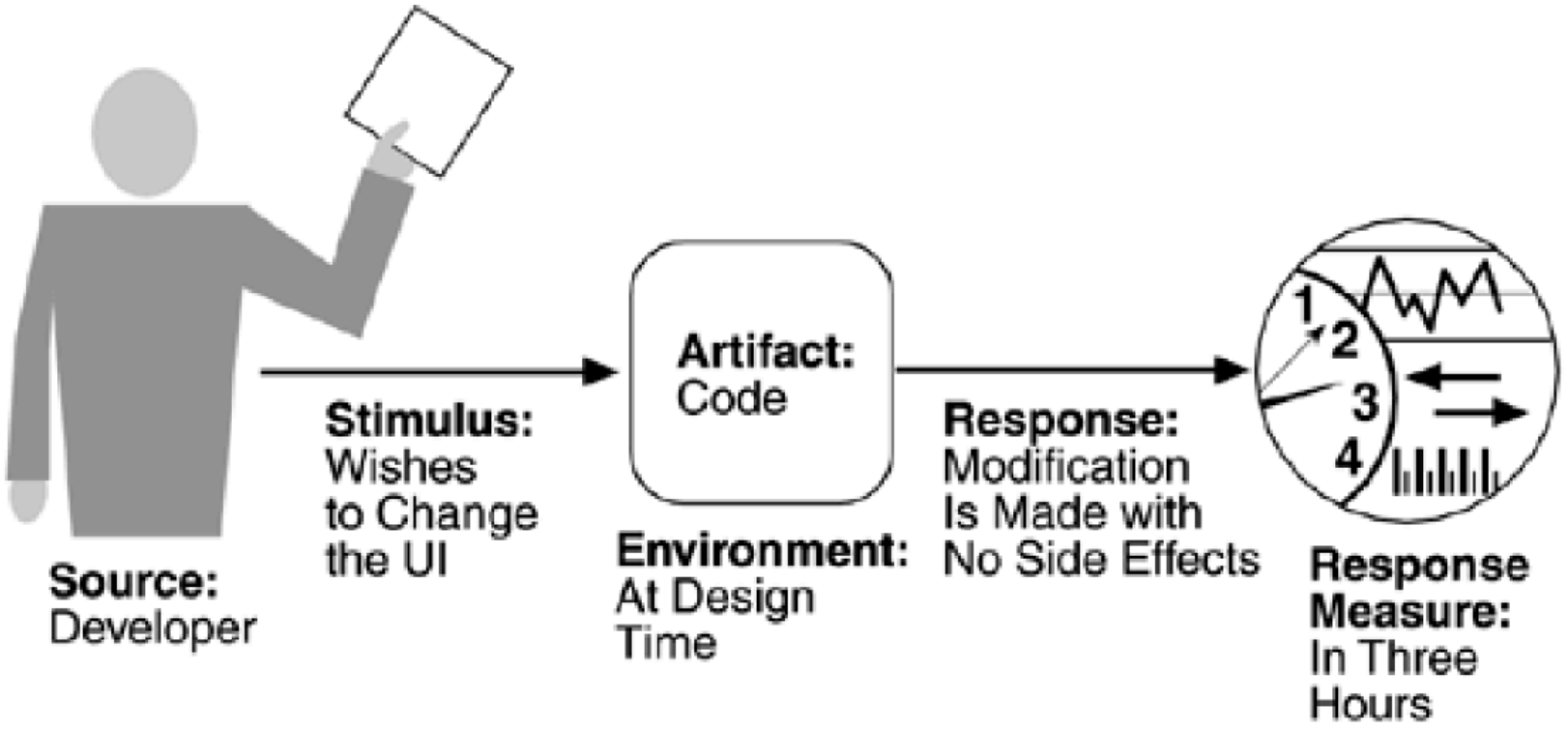
Quality Attribute Scenario- Availability



Read more: Software Architecture in Practice, By Len Bass, Paul Clements, Rick Kazman, Addison Wesley.

System Quality Attributes

Quality Attribute Scenario- Modifiability

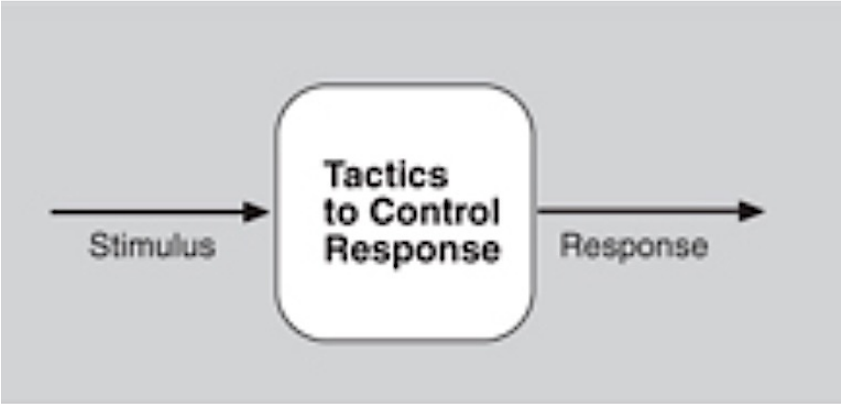


Read more: Software Architecture in Practice, By Len Bass, Paul Clements, Rick Kazman, Addison Wesley.

Architectural Tactics

- The achievement of quality attributes relies on fundamental design decisions.

- A *tactic* is a design decision that influences the achievement of a quality attribute response—tactics directly affect the system's response to some stimulus.



Architectural Tactics

- The focus of a tactic is on a single quality attribute response. Within a tactic, there is no consideration of tradeoffs.
- A collection of tactics is an architectural **strategy**.
- The tactics, similar to design patterns, are design techniques that architects have been using for years.
- The tactics needs to be refined by designers to make each tactic concrete.
  - The application of a tactic depends on the context.
    - *Manage sampling rate* is relevant in some real-time systems but not in all real-time systems and certainly not in database systems.

Availability

- Concerned with system failure and its associated consequences.
  - A system failure occurs when the system no longer delivers a service consistent with its specification.
  - Such a failure is observable by the system's users

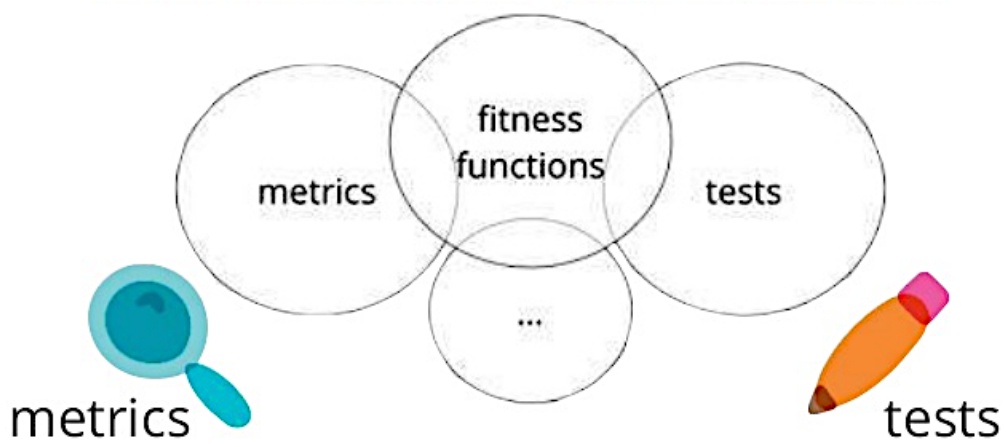
- Availability tactics
  1. will keep faults from becoming failures, or
  2. at least bounds the effects of the fault and make repair possible.

1. Fault Detection; recognizing fault
2. Fault Recovery; preparing for recovery and making the system repair
3. Fault Prevention

Architecture fitness function

Governing Architecture Characteristics

- Any mechanism that provides an objective integrity assessment of some architecture characteristic or combination of architecture characteristics



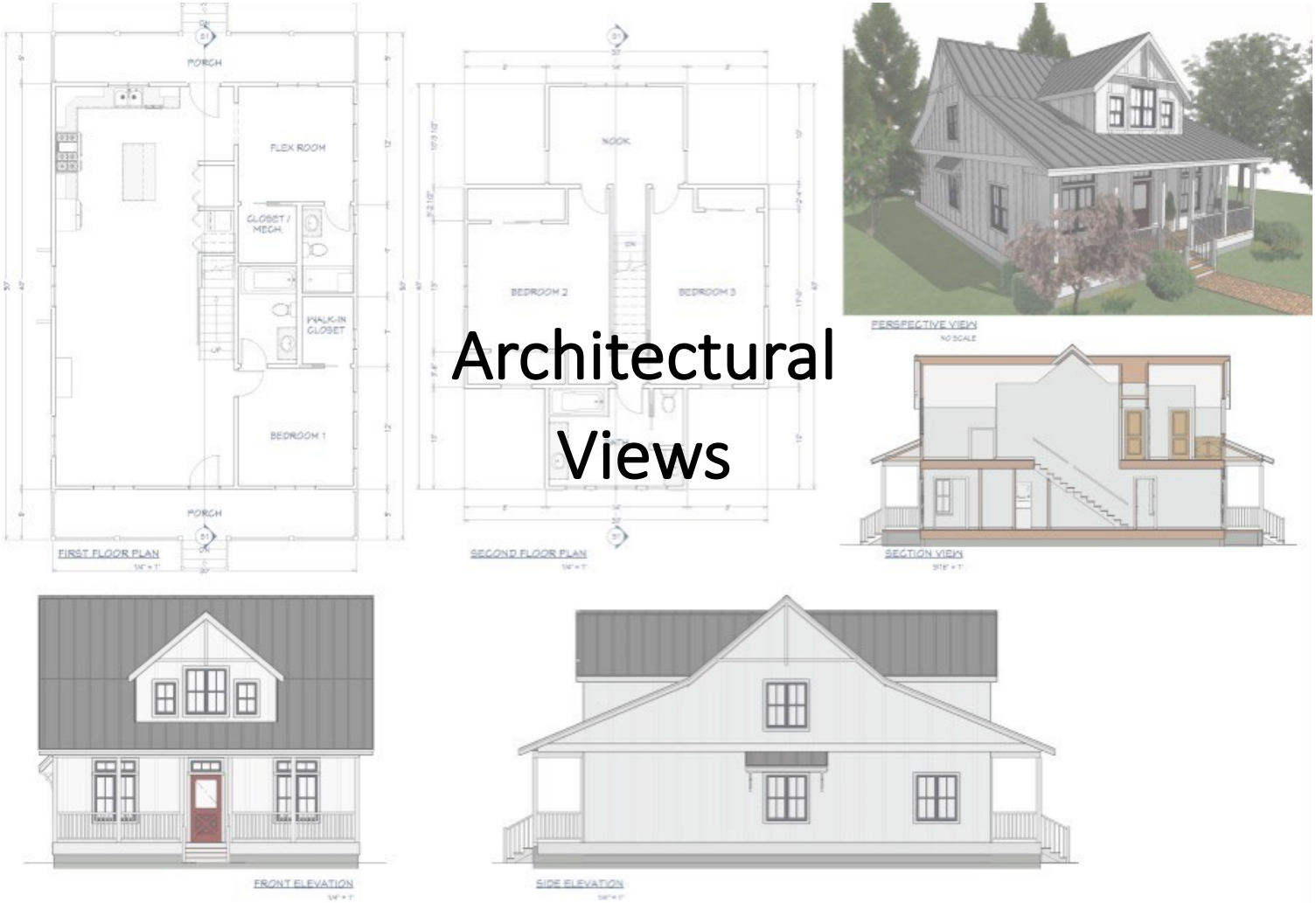
- **Modularity**
  - *Metric*: Cyclic dependencies
  - *Monitor*: write a fitness function to look after cycles

Diagramming and Presenting Architecture

"… effective communication becomes critical to an architect's success".

--Mark Richards

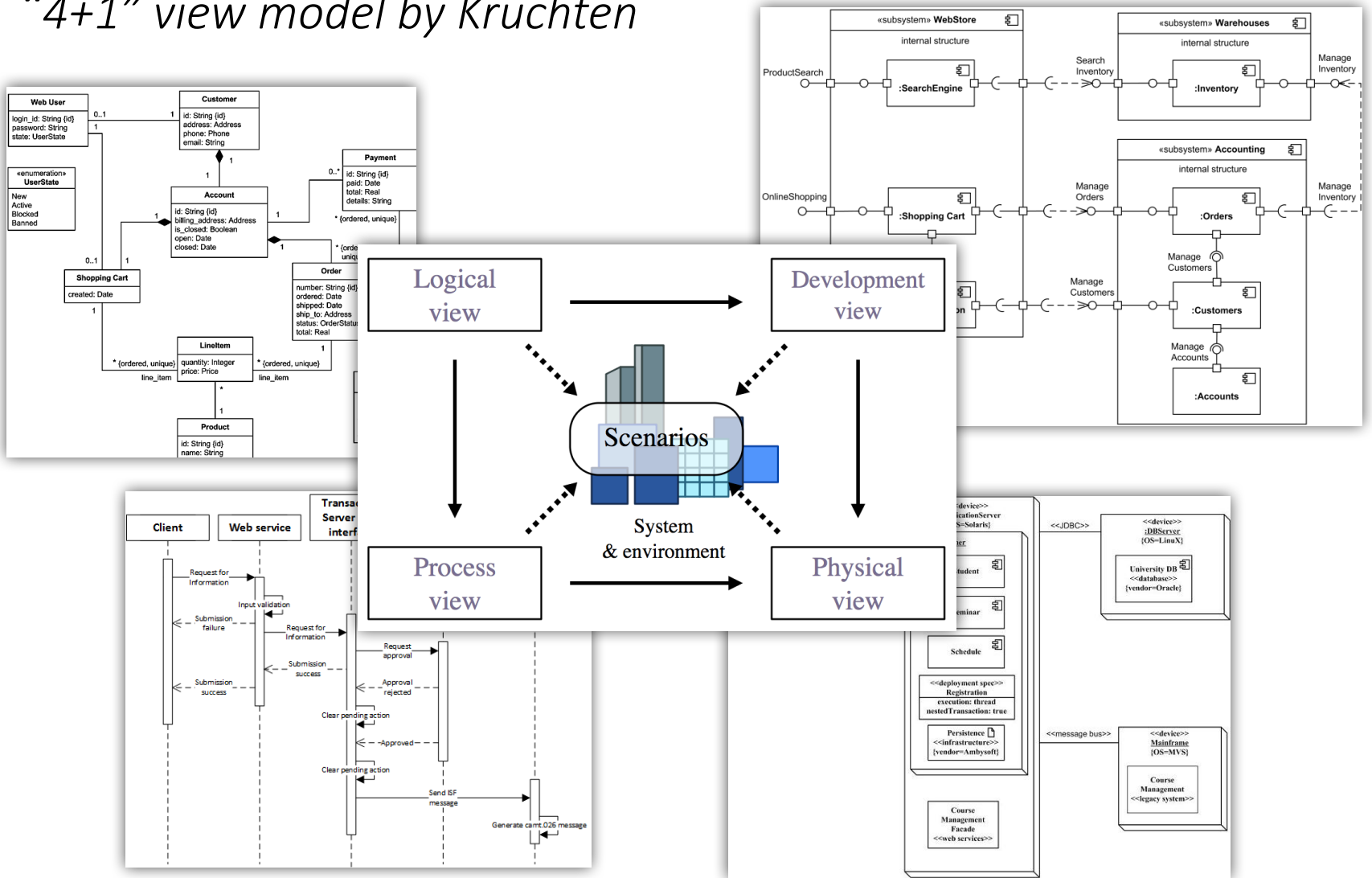
Architectural Views



A view is a **representation** of a coherent set of architectural elements, and their relations, from the perspective of a related set of **concerns**.

Architectural Views

"4+1" view model by Kruchten



Architectural Anti-Patterns

- Frozen Caveman
  - describes an architect who always reverts back to their pet irrational concern for every architecture
- Ivory Tower Architect
  - when architecture decisions are made isolated from the implementation team.
- Covering Your Assets
  - occurs when an architect avoids or defers making an architecture decision out of fear of making the wrong choice.
- Groundhog Day
  - occurs when people don't know why a decision was made, so it keeps getting discussed over and over and over.
- Email-Driven Architecture
  - where people lose, forget, or don't even know an architecture decision has been made.

For Further Reading!

- Len Bass, Paul Clements, Rick Kasman, *Software Architecture in Practice*, 3rd Edition, Addison-Wesley Professional, 2012.
- Clements, Bachmann, Bass , Garlan, Ivers, Little, Merson, Nord, and Stafford, *Documenting Software Architectures: Views and Beyond*, Addison-Wesley, Boston, MA, 2011.
- Robert Hanmer, *Pattern-Oriented Software Architecture FOR DUMMIES*, Wiley, 2013.
- Martin Fowler, *Microservices: A definition of this new architectural term*. 2014. Available in: http://martinfowler.com/articles/microservices.html.
- Mark Richards, *Software Architecture Patterns*, O'Reilly Media Inc, 2015. https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/
- Software Architecture in Practice: http://etutorials.org/Programming/Software+architecture+in+practice,+second+edition/