HCMC University of Technology, VNU

Software Architecture

Chapter 1. Introduction

Lecturer: Phat T. Tran-Truong, Meng.

Email: phattttt@hcmut.edu.vn

Aims

- Provide core concepts and techniques in **design principles**, design patterns, **architectural styles**, and architectural patterns.
- Learn to **make** informed **architectural decisions**, **evaluate trade-offs**, and **document designs**.
- Be capable of **analyzing**, **designing**, and **implementing** robust **software architectures**.

Outline

- Introduces the fundamental principles and practices of software architecture and design:
  - SOLID design principles
  - Key architectural concepts
  - Architectural styles
  - Documenting architectures using different views

Learning outcomes

No.	Learning Outcomes
L.O.1	**Understand the principles required for software design**
L.O.1.1	Demonstrate understanding of SOLID design principles
L.O.1.2	Be able to explain the notion of high cohesion & low coupling
L.O.2	**Be able to apply different software architecture styles**
L.O.2.1	Be able to apply monolithic architecture styles
L.O.2.2	Be able to apply distributed architecture styles
L.O.3	**Effectively apply architectural design tactics**
L.O.3.1	Analyze the impact of architectural characteristics on design decision-making
L.O.3.2	Analyze the trade-offs of software architectures
L.O.4	**Develop software architecture documents**
L.O.4.1	Present various design perspectives in the documentation
L.O.4.2	Use different types of diagrams to document software architecture

Textbook/reference book

- [1] Robert C. Martin, **Clean Architecture: A Craftsman's Guide to Software Structure and Design**, Pearson; 1st edition (September 10, 2017), ISBN 978-0134494166
- [2] Mark Richards, Neal Ford, **Fundamentals of Software Architecture: An Engineering Approach**, O'Reilly Media; 1st edition (January 28, 2020), ISBN: 978-1492043454
- [3] Paul Clements, Felix Bachmann, Len Bass, David Garlan, James Ivers, Reed Little, Paulo Merson, Paulo Merson, Robert Nord, Judith Stafford, **Documenting Software Architectures: Views and Beyond**, 2nd edition, 2011, Addison-Wesley Professional, ISBN 978-0321552686
- [4] Ian Sommerville. **Software Engineering**, 10th edition. Pearson Education, 2011.

Evaluation

- Group project: 40%
- Final exam: 60%
- Group seminar/In-class exercises/ homework: bonus

Contact

- Lecturers:
  - Dr. Thai-Minh Truong (Trương Thị Thái Minh) (thaiminh@hcmut.edu.vn)
  - Meng. Phat T. Tran-Truong (Trần Trương Tuấn Phát) (phatttt@hcmut.edu.vn)

- Course website:
  - https://lms.hcmut.edu.vn

Reference sources of the slides

- Slides in this course are adapted mainly from Richards et al. 2020 [4] and Martin 2017 [1].

[1] Robert C. Martin, Clean Architecture: A Craftsman's Guide to Software Structure and Design, Pearson; 1st edition (September 10, 2017), ISBN 978-0134494166
[4] Mark Richards, Neal Ford, Fundamentals of Software Architecture: An Engineering Approach, O'Reilly Media; 1st edition (January 28, 2020), ISBN: 978-1492043454

CHAPTER 1: INTRODUCTION

Chapter 1. Introduction

1.1. Overview of Design

1.2. Defining Software Architecture

1.3. Expectations of an Architect

1.4. Separation of Concerns

1.5. The Intersection of Software Architecture with DevOps, Processes, and Data

1.1. Overview of Design

"The goal of software architecture is to minimize the human resources required to build and maintain the required system"

"The measure of design quality is simply the measure of the effort required to meet the needs of the customer."

=>*"If that effort is low, and stays low throughout the lifetime of the system, the design is good."*

1.1. Overview of Design

"In most successful software projects, the expert developers working on that project have a shared understanding of the system design. This shared understanding is called architecture."

— Martin Fowler

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

ANSI/IEEE Std 1471-2000

"The software architecture of a program or computing system is the ***structure or structures*** of the system, which comprise ***software elements***, the ***externally visible properties*** of those elements, and the ***relationships*** among them."

SEI- Software Engineering Institute

"[Software architecture goes] beyond the algorithms and data structures of the computation; designing and specifying the ***overall system structure*** emerges as a new kind of problem. ***Structural issues*** include gross organization and global control structure; ***protocols for communication, synchronization, and data access; assignment of functionality to design elements; physical distribution; composition of design elements; scaling and performance;*** and ***selection among design alternatives***."

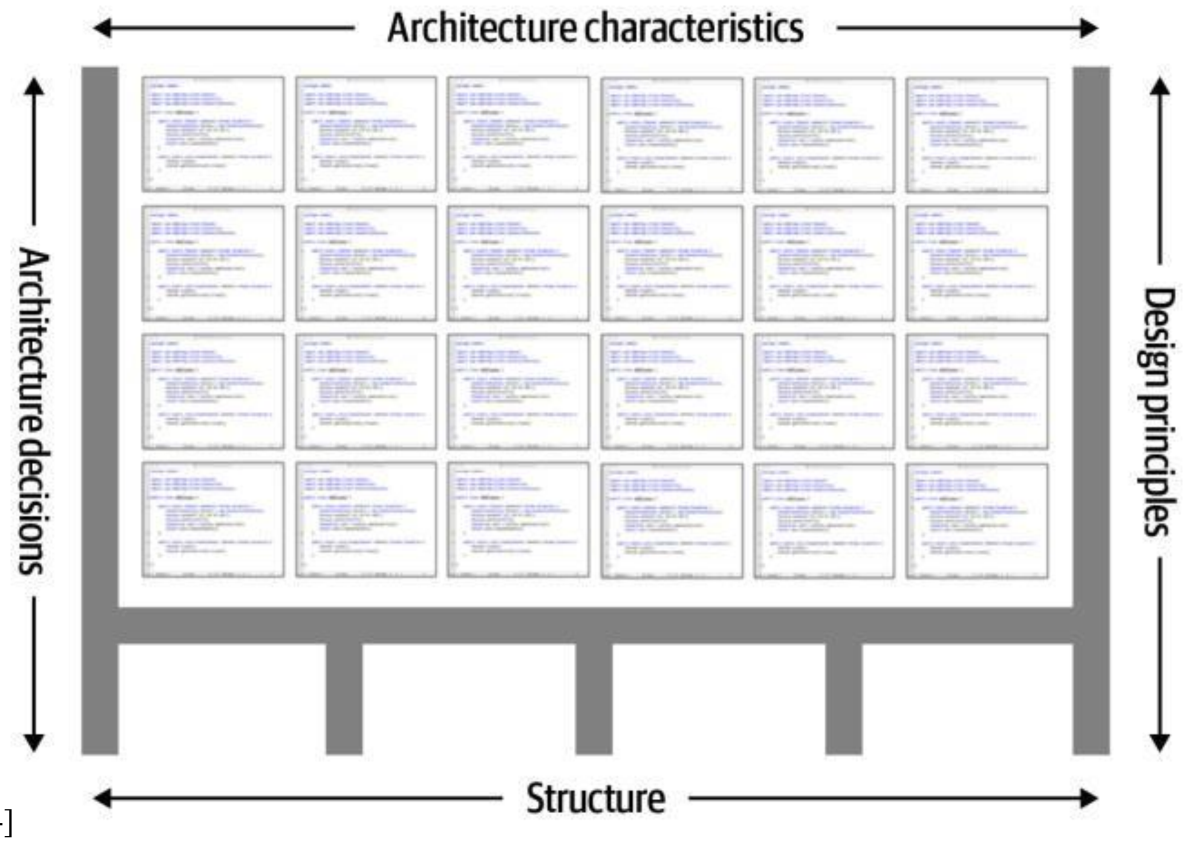
Garlan and Shaw

●●●

1.2. Defining Software Architecture

Four dimensions that define software architecture

- Software architecture consists of the structure of the system, combined with architecture characteristics (“-ilities”) the system must support, architecture decisions, and finally design principles.

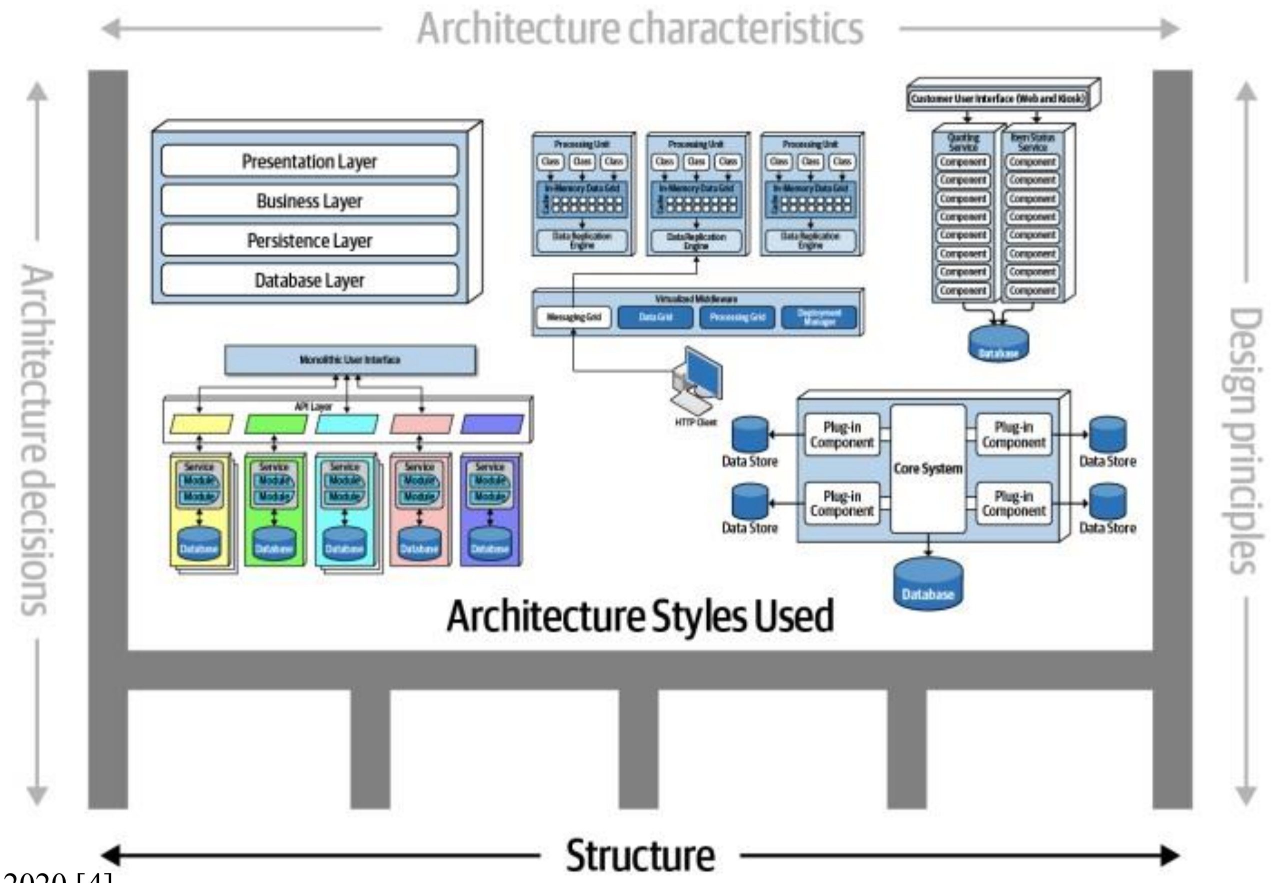


Richards et al. 2020 [4]

1.2. Defining Software Architecture

Structure of the system

- Structure refers to the type of architecture styles used in the system such as microservices, layered, or microkernel

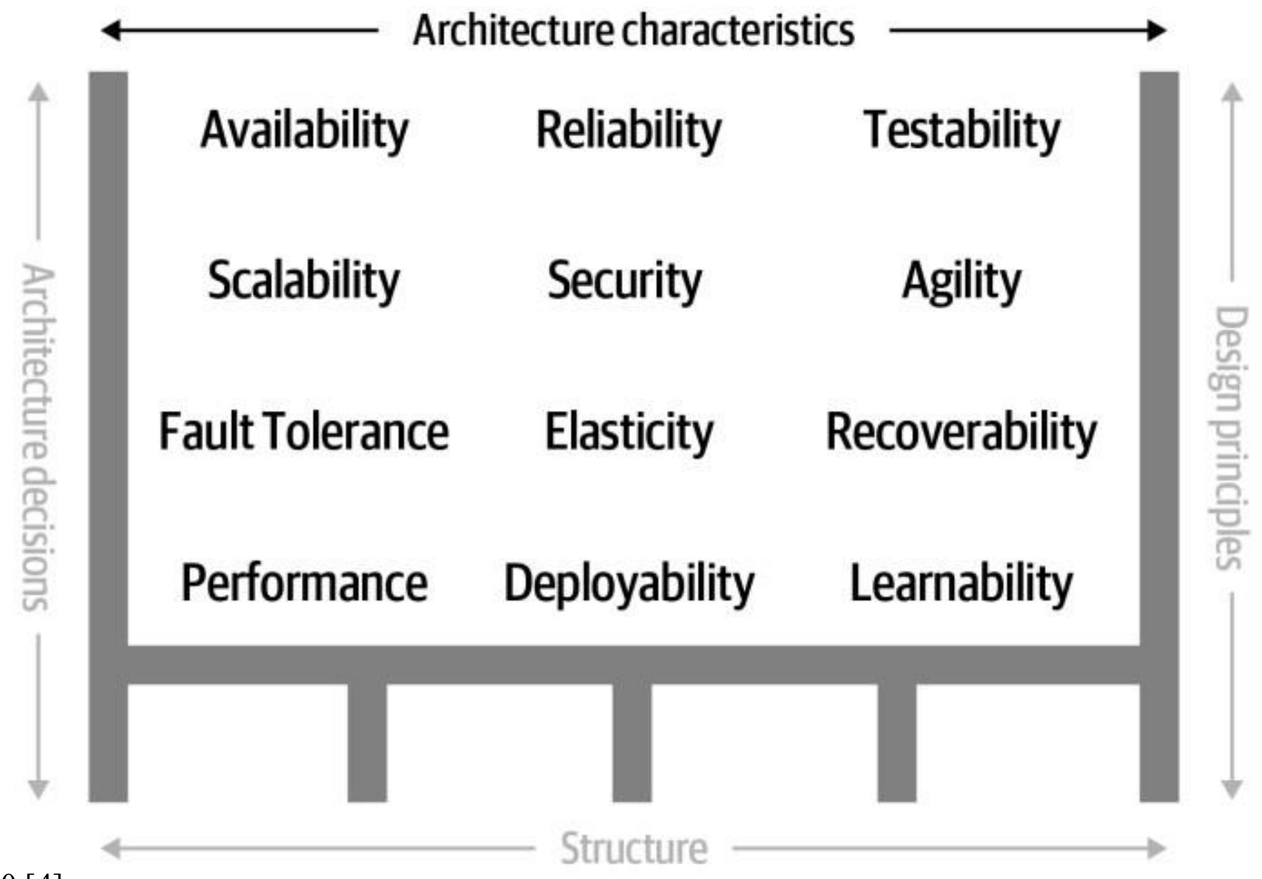


Richards et al. 2020 [4]

1.2. Defining Software Architecture

Architecture characteristics

- The architecture characteristics define the success criteria of a system, which is generally unrelated to the functionality of the system.

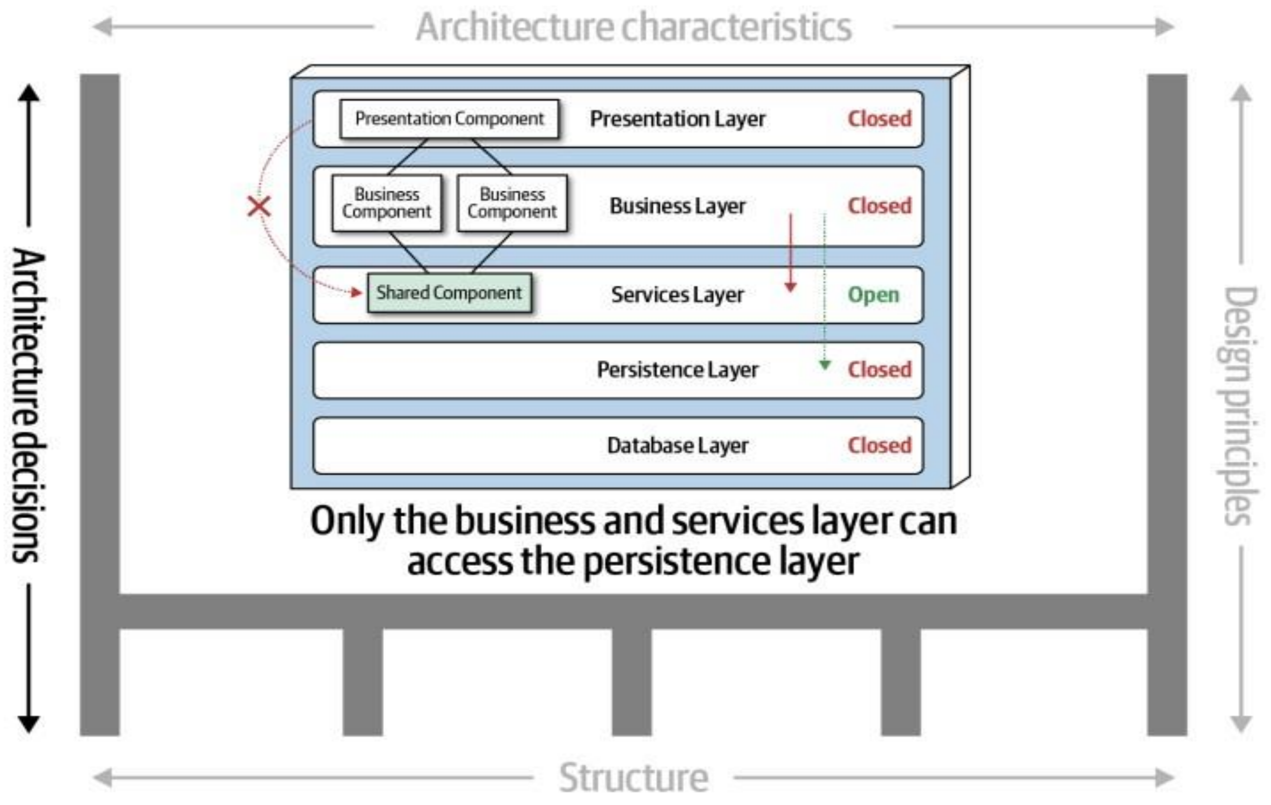


Richards et al. 2020 [4]

1.2. Defining Software Architecture

Architecture decisions

- Architecture decisions define the rules for how a system should be constructed. Architecture decisions form the constraints of the system and direct the development teams on what is and what isn't allowed.

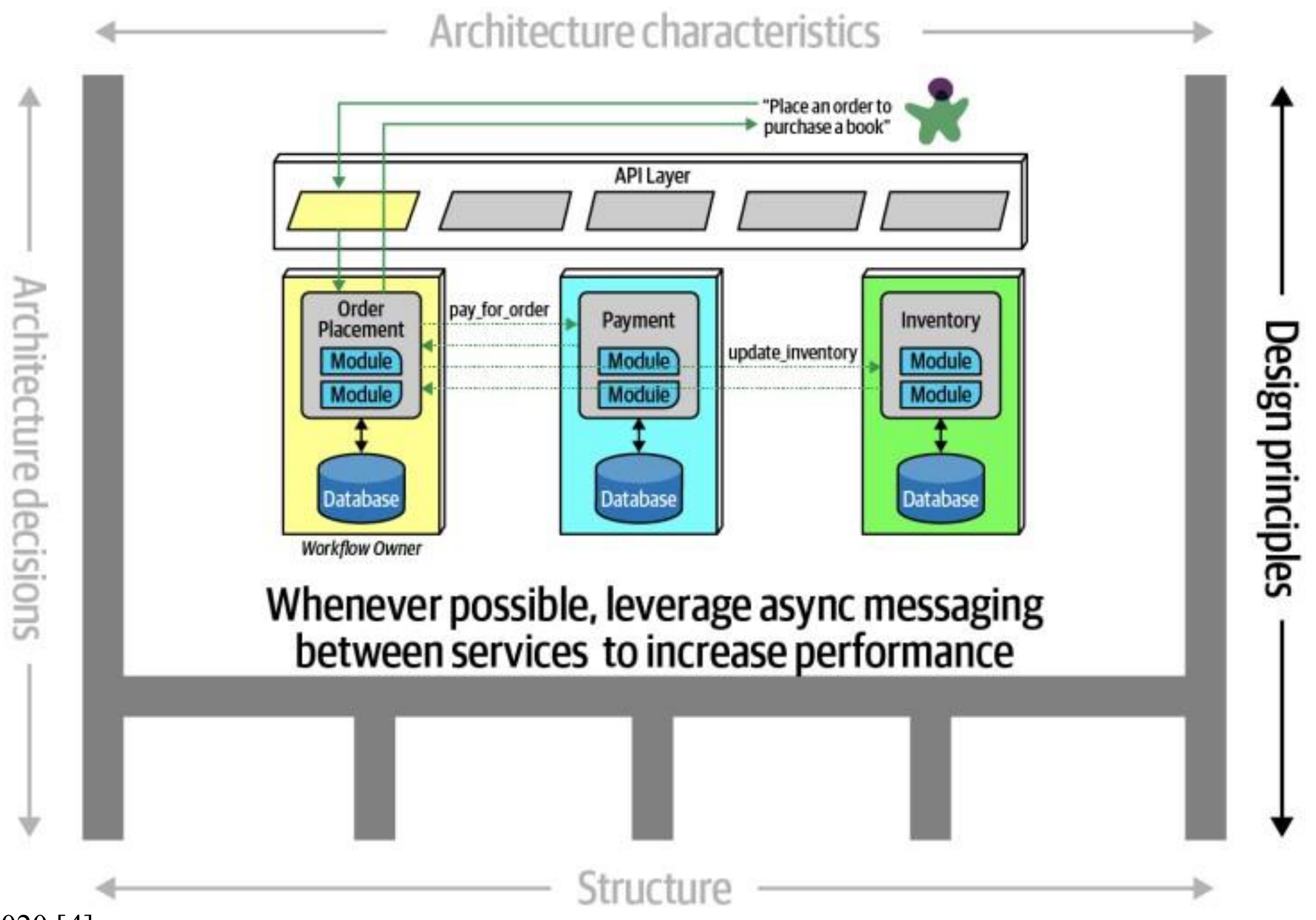


Richards et al. 2020 [4]

1.2. Defining Software Architecture

Design principles

- Guidelines for constructing systems



Richards et al. 2020 [4]

1.2. Defining Software Architecture

Architecture Decision vs. Design Principle

- A design principle differs from an architecture decision in that a **design principle is a guideline rather than a hard-and-fast rule**.
- If a particular architecture decision cannot be implemented in one part of the system due to some condition or other constraint, that decision (or rule) can be broken through something called a **variance**.

- *<u>Example</u>: An architecture decision (rule) could never cover every condition and option for communication between services, so a design principle can be used to provide guidance for the preferred method (in this case, asynchronous messaging) to allow the developer to choose a more appropriate communication protocol (such as REST or gRPC) given a specific circumstance.*

1.3. Expectations of an Architect

Core expectations of a software architect

1. Make architecture decisions
2. Continually analyze the architecture
3. Keep current with latest trends
4. Ensure compliance with decisions
5. Diverse exposure and experience
6. Have business domain knowledge
7. Possess interpersonal skills
8. Understand and navigate politics.

SA - Laws of Software Architecture

Laws of Software Architecture

First Law

- Everything in software architecture is a trade-off.
  - **Corollary 1:** If an architect thinks they have discovered something that isn't a trade-off, more likely they just haven't identified the trade-off yet.

Second Law

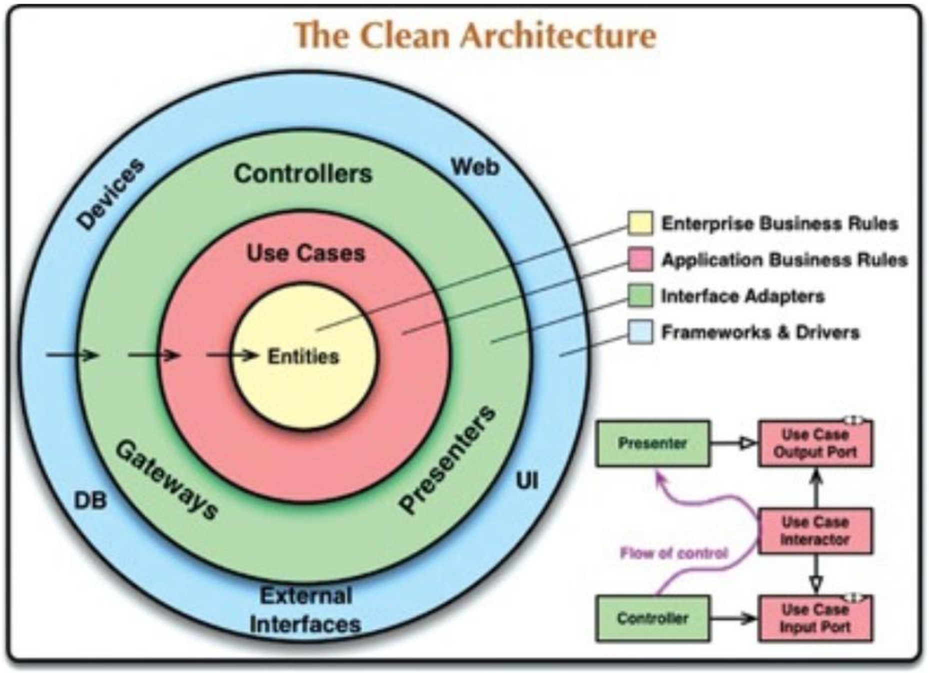
- WHY is more important than HOW.

1.4. Separation of Concerns

- Although software architectures all vary some what in their details, they are very similar.
- They all have the same objective, which is the separation of concerns.
- They all achieve this separation by dividing the software into layers. Each has at least one layer for business rules, and another layer for user and system interfaces.

1.4. Separation of Concerns

- Each of these architectures produces systems that have the following characteristics:
  - *Independent of frameworks*
  - *Testable.*
  - *Independent of the UI*
  - *Independent of the database.*
  - *Independent of any external agency.*



Martin 2017 [1]

1.5. The Intersection of Software Architecture with DevOps, Processes, and Data

Operations/DevOps

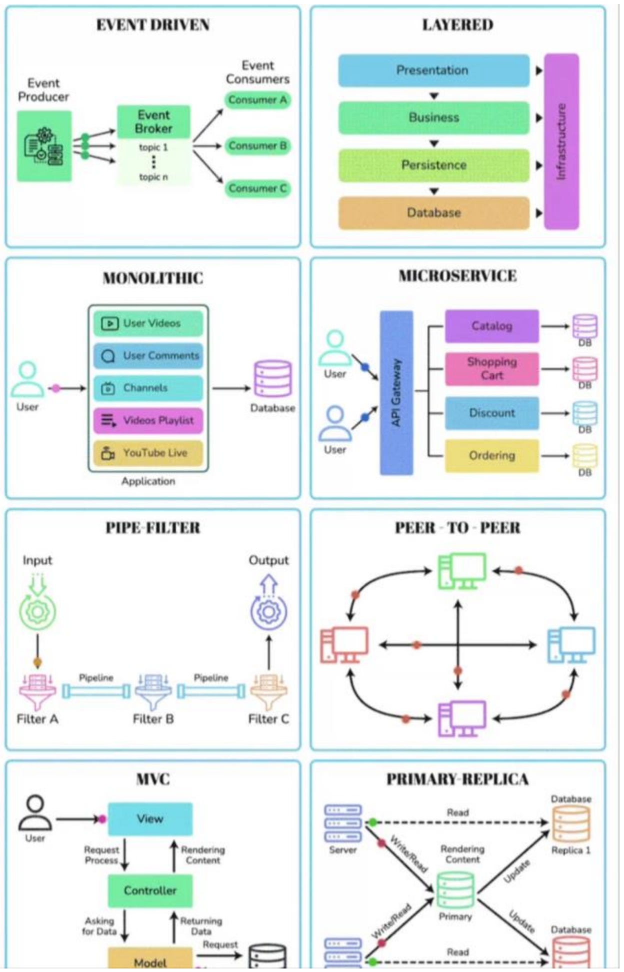
- The most obvious recent intersection between architecture and related fields occurred with the advent of DevOps,
- Many architectures designed during the 1990s and 2000s assumed that architects couldn’t control operations
- New forms of architecture that combine many operational concerns with the architecture
  - ESB-driven SOA, Microservices

Process

- Software architecture is mostly orthogonal to the software development process
- The way that you build software (process) has little impact on the software architecture (structure)
- However, the process by which teams develop software has an impact on many facets of software architecture
  - Architects in Agile projects

Data

- A large percentage of serious application development includes external data storage, often in the form of a relational (or, increasingly, NoSQL) database.
- Database administrators often work alongside architects to build data architecture for complex systems, analyzing how relationships and reuse will affect a portfolio of applications.


EVENT DRIVEN
Event Producer
Event Broker
topic 1
topic n
Event Consumers
Consumer A
Consumer B
Consumer C
LAYERED
Presentation
Business
Persistence
Database
Infrastructure
MONOLITHIC
User
User Videos
User Comments
Channels
Videos Playlist
YouTube Live
Application
Database
MICROSERVICE
User
User
API Gateway
Catalog
Shopping Cart
Discount
Ordering
DB
PIPE-FILTER
Input
Output
Pipeline
Pipeline
Filter A
Filter B
Filter C
PEER - TO - PEER
MVC
User
View
Request Process
Rendering Content
Controller
Asking for Data
Returning Data
Request
Model
PRIMARY-REPLICA
Server
Read
Write/Read
Rendering Content
Primary
Update
Database
Replica 1
Read
Database

Q&A