Clean Architecture

CompFest SEA 2021

by Aldo

```
$ whoami
LINE ID Software Engineer

$ ./software-architecture-experiences
- 2015-2017 no architecture
- 2017-2018 use clean architecture in several Android projects
- 2018-2020 no architecture
- 2020-2021 created two architectures

[WARNING] this presentation is based on my limited experience
```

Agenda:

- Software architecture
- Clean architecture
- Hexagonal architecture
- Summary
- (Hands-on) How to build your own architecture

Software Architecture

Software Architecture

- Code structures, components, and it's relations
- Blueprint
- Fundamental choices (cheap and expensive choices)

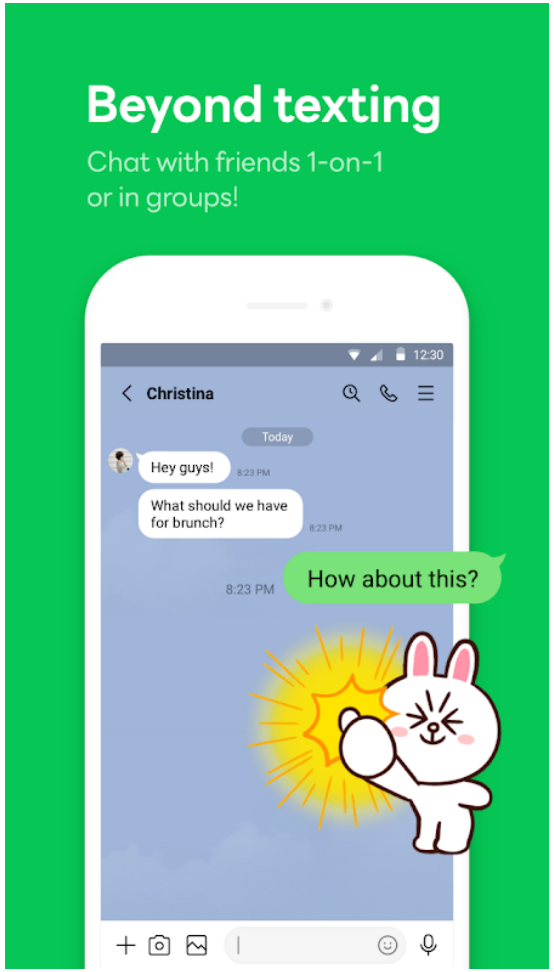
Why use architecture?

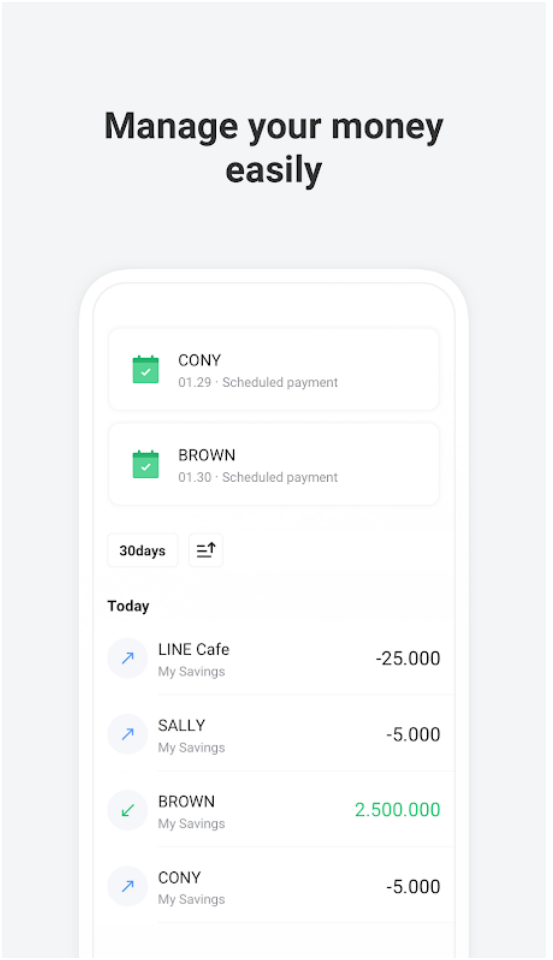
Why not?

- Every project has an architecture: By coincidence or by design
- But, architecture by coincidence is not scalable
- Do you need scalability?
  - Project lifetime
  - People involved

Signs of scalability issues:

- Hard to verify important logic
  - Hard to test





Signs of scalability issues:

- Hard to add new things
  - Where to put things?
  - Breaking changes anxiety
- You can't onboard new engineers easily
  - No clear organization

Why use architecture?

- You can focus on what matters
  - Is testability important to you?
  - How about build speed, responsiveness, reliability?
  - Portability and time-to-market?

Why use architecture?

- The code is more understandable
  - Human is just good at pattern recognition
  - Improve communication

When should I use architecture?

- It's not I, it's "we"
  - Architecture is a team decision
  - Architecture spread across the codebase
    - All team members must maintain the architecture
  - In some cases, even outside of the dev team

When should we use architecture?

- When we're having a scalability issue
  - Better: When we know, we WILL have a scalability issue
- You and your team know what's the best for your code

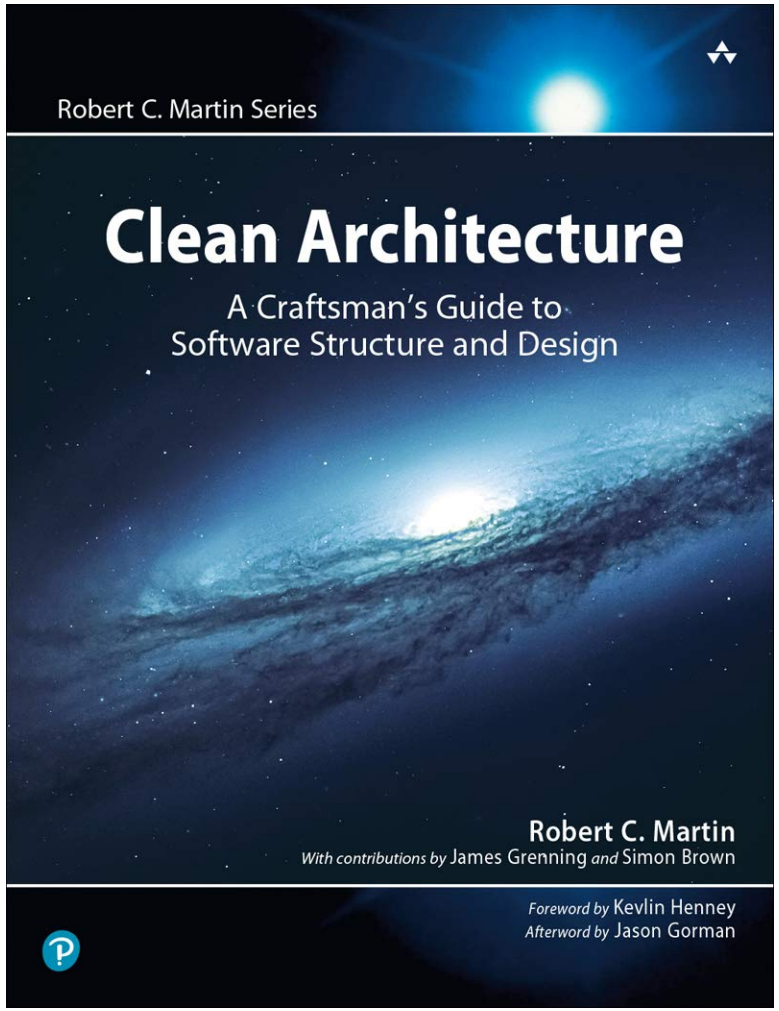
Software architecture examples

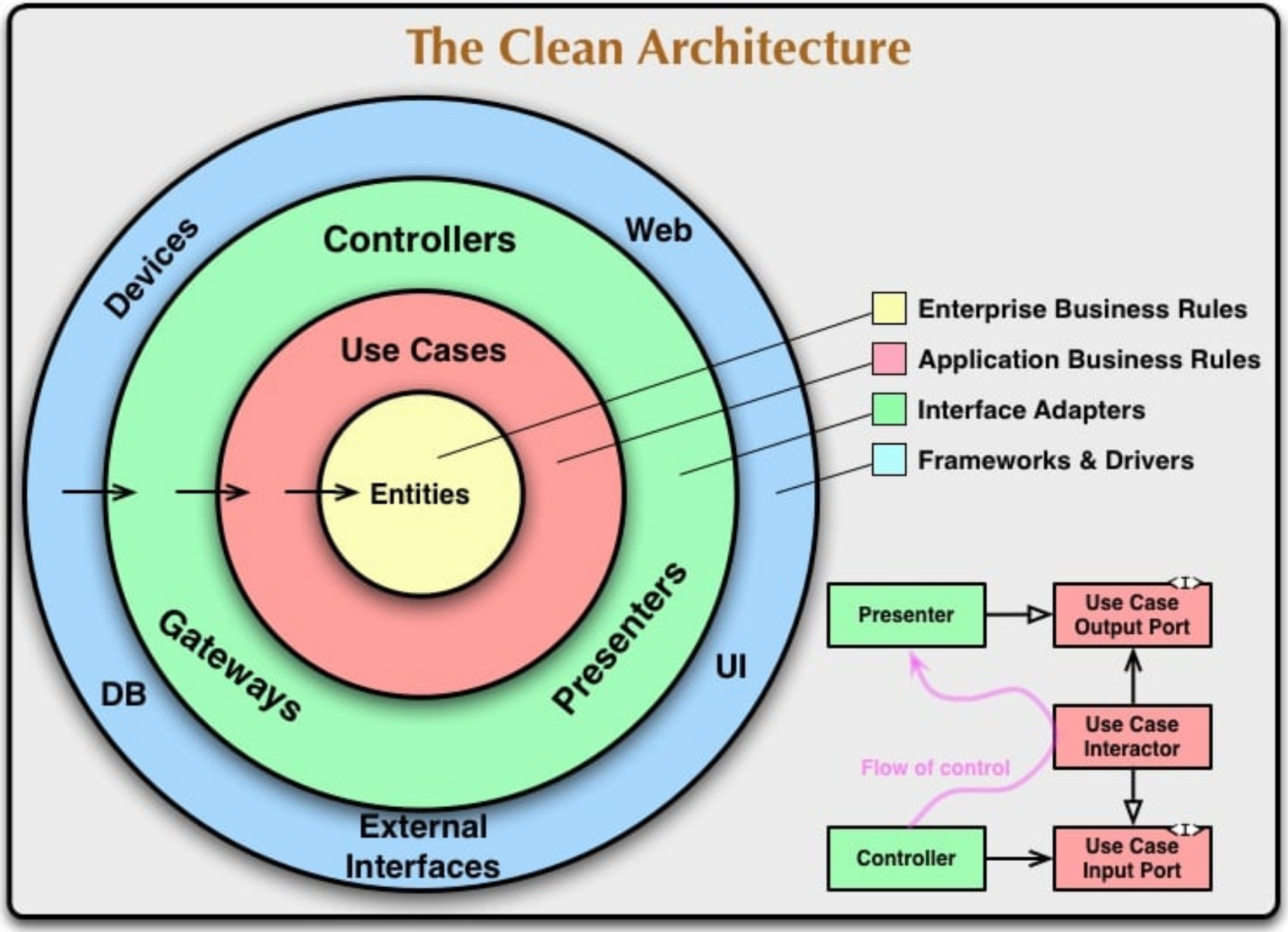
- Layered architecture
  - Presentation, Service, Persistence
- Presentation (UI) architecture:
  - MVC, MVVM, MVP
- Clean architecture
- (Insert your own architecture)

Clean Architecture

What is Clean Architecture?

- "Invented" by Bob C. Martin (Uncle Bob)
  - Published as a blog post at 13 Aug 2012
  - And as a book on 2017




The Clean Architecture
Devices
Controllers
Web
Use Cases
Entities
Gateways
Presenters
DB
UI
External
Interfaces
Enterprise Business Rules
Application Business Rules
Interface Adapters
Frameworks & Drivers
Presenter
Use Case
Output Port
Use Case
Interactor
Flow of control
Controller
Use Case
Input Port

What is Clean Architecture?

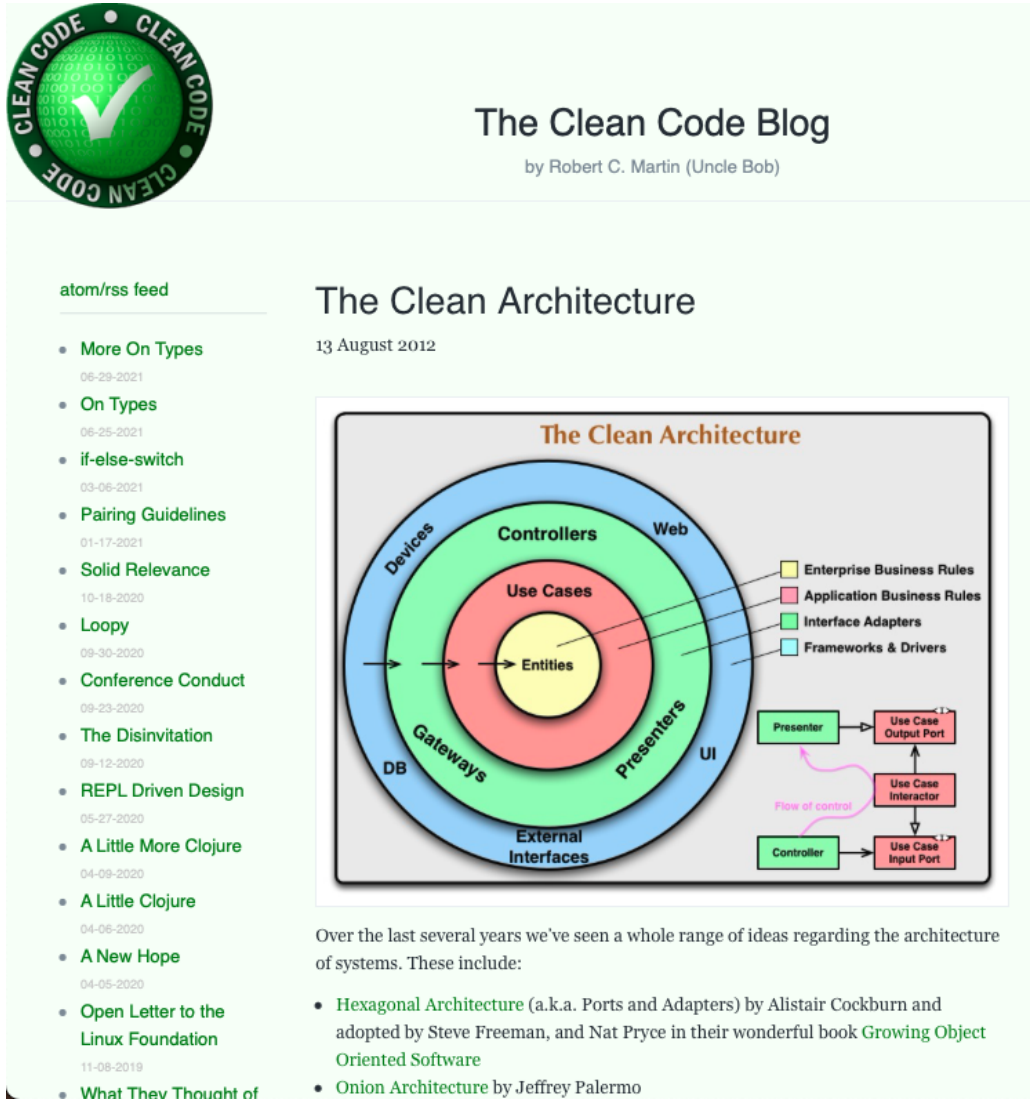
- Based on other existing architectures
  - Hexagonal, Onion, etc.
  - Focused on dependency rule
- Architecture that put **business rule** at the core
  - a.k.a Domain/Use Case/Feature
  - Everything else is secondary

Why Clean Architecture?

- Portable
  - Independent of Frameworks (UI, DB, others)
- Testable
  - The business rules are testable

Purist Approach to Clean Architecture

- Concept, not implementation
  - Not much code sample and diagram
- Blog post as the source of truth
- Feel free to build your own implementation

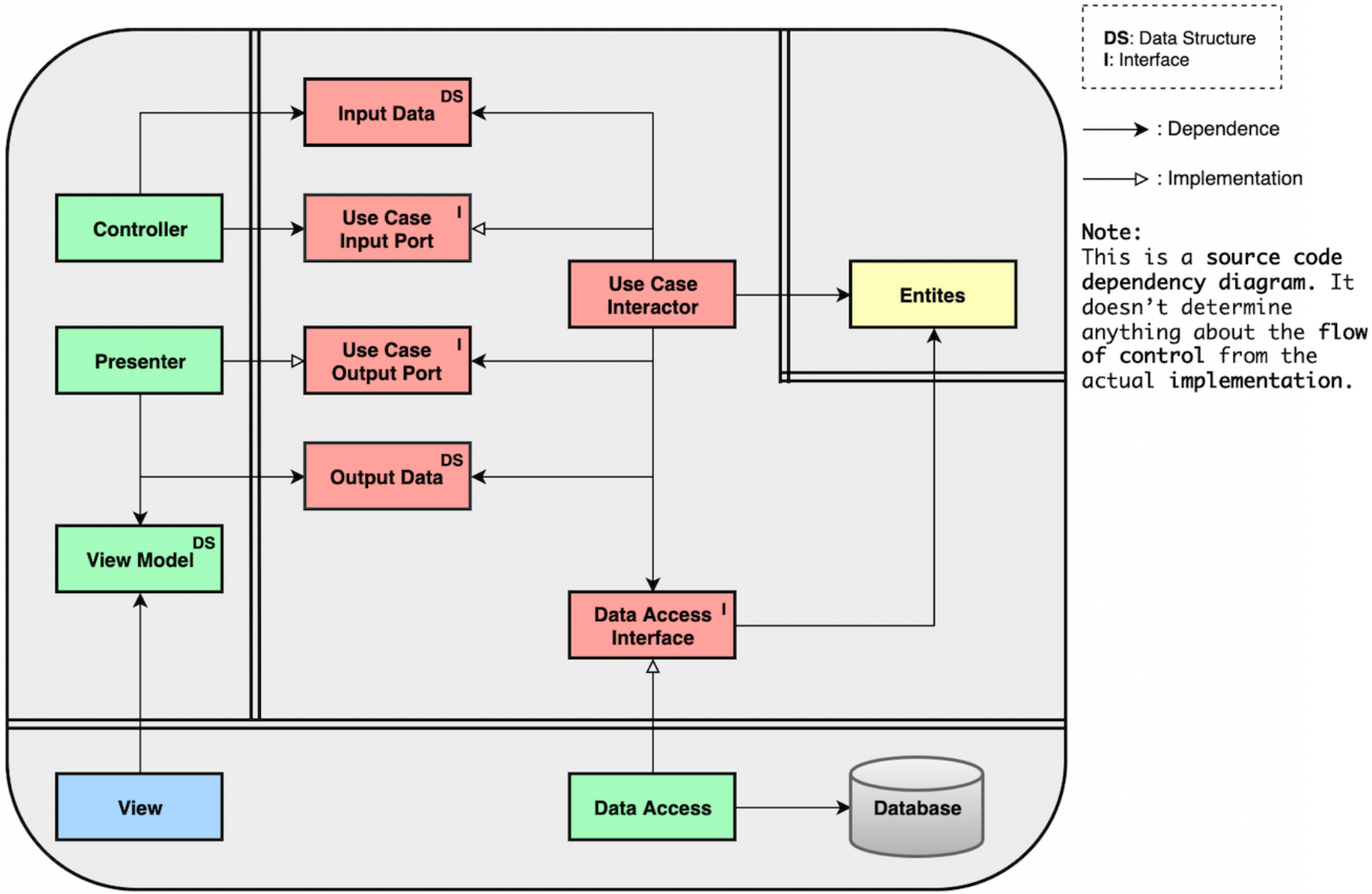


Official Layers in Clean Architecture

- Enterprise Business Rules
  - Entities
- Application Business Rules
  - Use Cases
- Interface Adapters
  - Controllers, Gateway, Presenters
- Framework and Drivers
  - Devices, Web, UI, DB

Demo: Get Article

- "Hello world" for clean architecture
- Get an article
  - No more, no less (proved by test)
- `clean-architecture-java` as a "server" code


DS
Input Data
Controller
Use Case
Input Port
I
Use Case
Interactor
Entites
Presenter
Use Case
Output Port
I
DS
Output Data
DS
View Model
Data Access
Interface
I
View
Data Access
Database
DS: Data Structure
I: Interface
: Dependence
: Implementation
Note:
This is a source code dependency diagram. It doesn't determine anything about the flow of control from the actual implementation.

Layers in Clean Architecture (Recap)

- Entities
  - Unlikely to change. The actor in the application
  - POJO, PORO, Data class
- Use case
  - Control the **dance** of entities
  - Interface/Boundary to the outside layer

Layers in Clean Architecture (Recap)

- Adapters
  - Interface/Boundary implementation
  - Spring's Controller, Android's Activity, etc.
- Framework
  - Framework-only code
  - Main class, config class, etc.

Demo: Loan Application

- Apply for a loan using a console
- Example from Clean Architecture book
- `clean-architecture-loan-application` as a console application

Hexagonal Architecture

Hexagonal Architecture

- Created by Alister Cockburn on January 4, 2005
  - The blog post
- Blog post as the source of truth
- 🎬 Netflix use this!

The Pattern: Ports and Adapters ("Object Structural")

Alternative name: "Ports & Adapters"

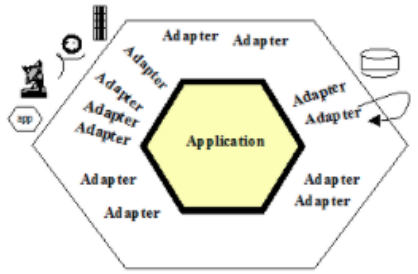
Alternative name: "Hexagonal Architecture"

Intent

Allow an application to equally be driven by users, programs, automated test or batch scripts, and to be develo isolation from its eventual run-time devices and databases.

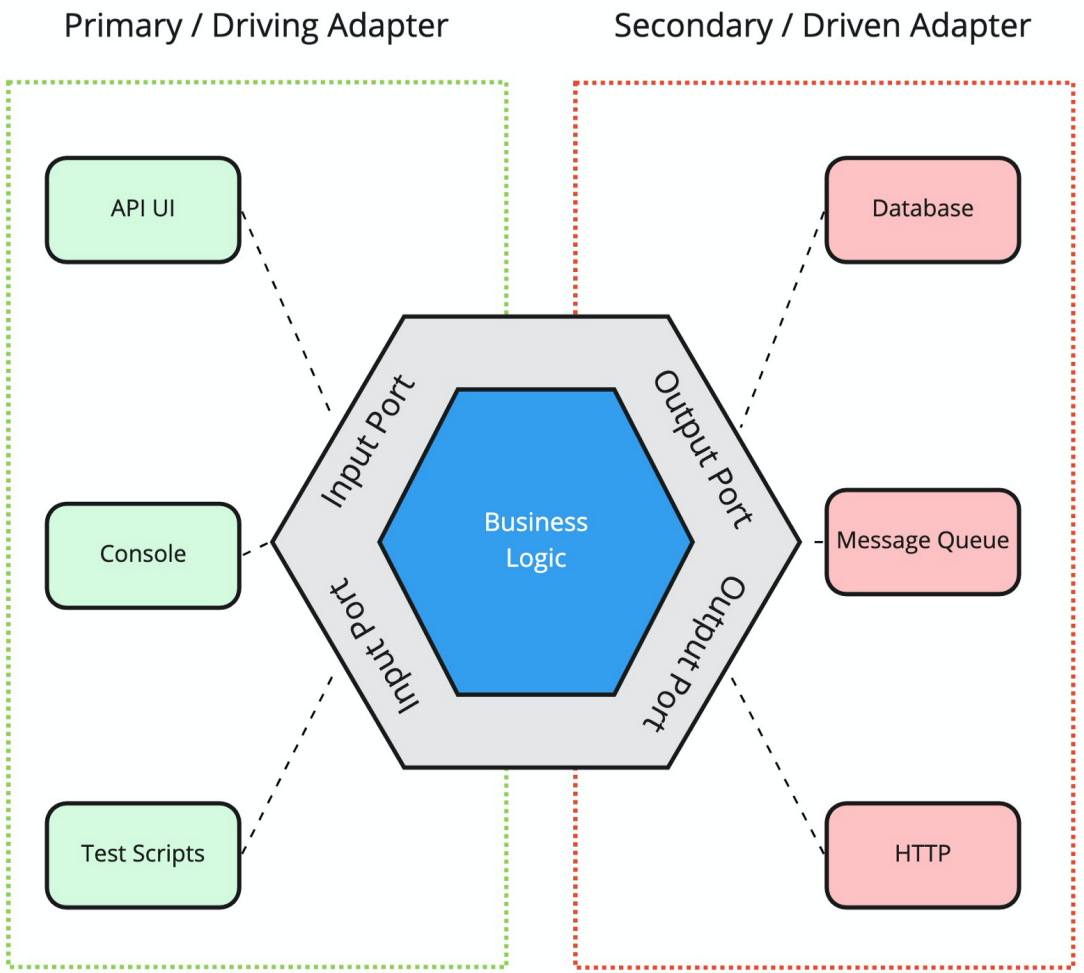
As events arrive from the outside world at a port, a technology-specific adapter converts it into a usable proced and passes it to the application. The application is blissfully ignorant of the nature of the input device. When th something to send out, it sends it out through a port to an adapter, which creates the appropriate signals need technology (human or automated). The application has a semantically sound interaction with the adapters on a without actually knowing the nature of the things on the other side of the adapters.

Figure 1 : Hexagonal architecture basic.gif



Layers in Hexagonal Architecture

- Application
- Port
- Adapter



Layers in Hexagonal Architecture

- Application -> Entity
- Port -> Use case
- Adapter -> Adapter

Demo: Tax Saver

- To save and calculate tax
- Five adapters
  - Input: LINE Bot, Telegram Bot, HTML
  - Output: HashMap and Firebase Firestore
- `hexagonal-architecture-tax-saver` as a Spring boot app

Summary

- Every architecture has pros and cons
- Find the architecture that suits your needs!

Assets

- Presentation and sample codes https://github.com/line-indonesia/compfest-sea-clean-architecture

Extra sample codes

- `clean-architecture-java` as a "server" code
- `clean-architecture-stereotype`
  - More detail in Get Article
  - Dependency inversion
- `command-pattern`
  - Recurring pattern in clean arch
- `progressive-tax.py`
  - Unit test demo in Python

How to Build your Own Architecture

Agenda

- How to build your own architecture
- Discussion with your team (30 mins)
- Presentation (2 mins/team)

How to Build your Own Architecture

- Find your goal
  - Tech stack, speed, flexibility, portability
- Find reference
  - The best practice
- Prototype & discuss
- Execute & evaluate

Modular Architecture (Android app)

- Goal
  - Build speed (Gradle parallelization)
  - Testable
  - Independence from server API

Modular Architecture (Android app)

- Result
  - Not Clean Architecture
  - No layer between MVVM and Repository
  - 70% speed reduction (from 155s to 43s)

Discussion with your Team (30 mins)

Output

- Text description
- Code
  - Snippet/Pseudocode
  - Proof of concept
- (Optional)
  - Diagram/Images
  - Reference(s)

Presentation (2 mins/team)

Resources

- https://play.google.com/store/apps/details?id=jp.naver.line.android
- https://play.google.com/store/apps/details?id=id.co.linebank
- https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html
- https://alistair.cockburn.us/hexagonal-architecture/
- https://www.amazon.com/Clean-Architecture-Craftsmans-Software-Structure/dp/0134494164
- https://medium.com/idealo-tech-blog/hexagonal-ports-adapters-architecture-e3617bcf00a0