LECTURE 3: LAYERING & MODULARITY

ANDREY.BOCHARNIKOV@GMAIL.COM

TELEGRAM: @RICKO_X

GOALS OF ARCHITECTURE

- Reduce complexity
- Evolutionary development of the system
- Common concept system

GOOD ARCHITECTURE

- Support use cases
- Maintenance, development, deployment
- Low coupling high cohesion
- Don’t repeat yourself

USE CASES

- Support intent of the system
- Separate details from policy
- Decouple policy from the details. Policy shouldn't depend on the details

EFFECTIVENESS

- Requirements
  - For example support 100k requests per second

- Monolithic program
- Micro-services run in parallel
- Multiple threads

CONWAY'S LAW, DEPLOYMENT&DEVELOPMENT

- "Any organization that designs a system will produce a design whose structure is a copy of the organization's communication structure."

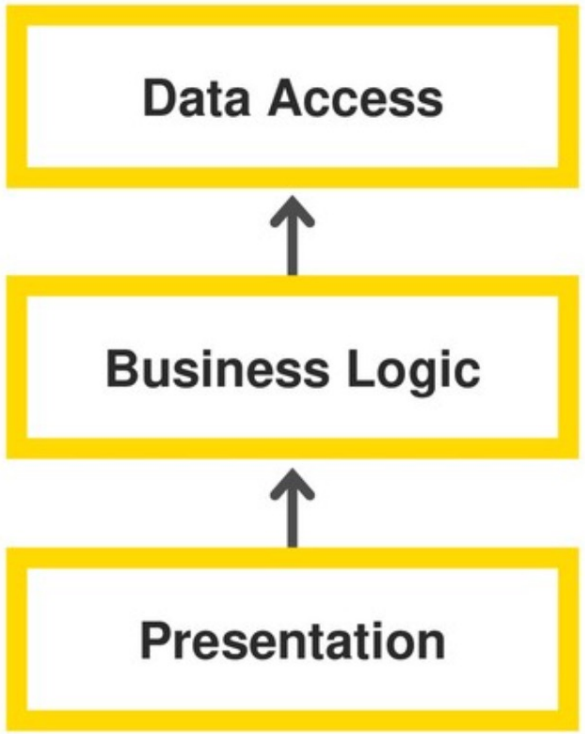
- Teams do not interfere with each other and can work independently.
- Immediate deployment after build.

BALANCE

- Architecture supports
  - The use cases and operation of the system
  - The maintenance of the system
  - The development of the system
  - The deployment of the system
- Leaving options open

LAYERS

- Single responsibility principle
- Separate those things that change for different reasons
- Collect those things that change for the same reasons



USE CASES

- Separate the system in to vertical slices to separate UI, Business rules and database for different use cases

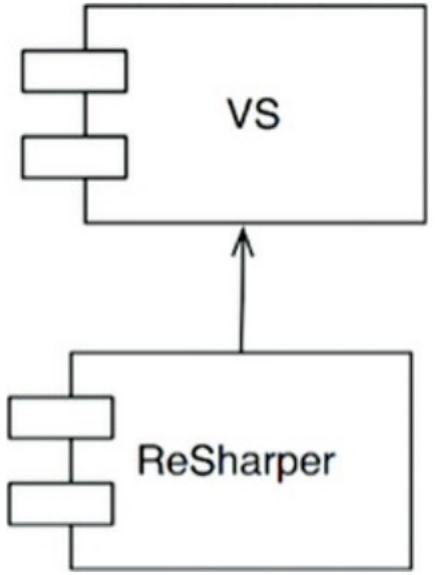
DECOUPLING MODES

- Source code level
- Deployment level
- Service level

- As the project evolves, the optimal mode may change.

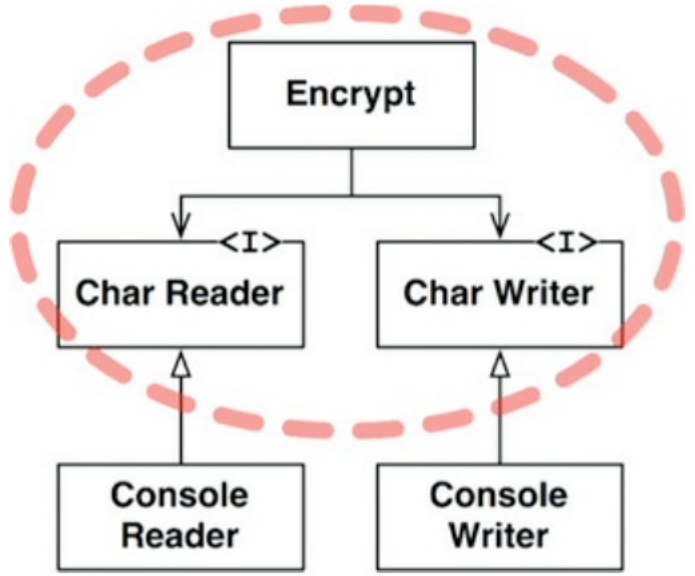
DEPENDENCY INVERSION

- Partition the system into components
- Find core business rules, plugins



LAYER LEVEL

- Policies
  - Describe how particular business rules are to be calculated
  - Input validation
  - Output formatting
  - …
- Level" is "the distance from the inputs and outputs



EXAMPLE

```
def encrypt():
  while True:
    writeChar(translate(readChar()))
```