Adapt or perish, now as ever, is nature's inexorable imperative.

- H.G. Wells

Modifiability Layered Pattern

WEEK 1-2

Today

Modifiability

The Layered Pattern

Change happens!

- To add new feature, to retiree old ones
- To fix defects, tighten security, or improve performance
- To enhance new protocols, new standards
- To make systems work together, even if they were never designed to do so

What is the cost of change?

Two types of cost:

- The cost of introducing the mechanism to make the system more modifiable
- The cost of making the modification using the mechanism

Justification for a Change

For **N** similar modification,

C_n : Cost of making the change without the mechanism,
C_w: Cost of making the change with the mechanism, and
C_i: Cost of installing the mechanism

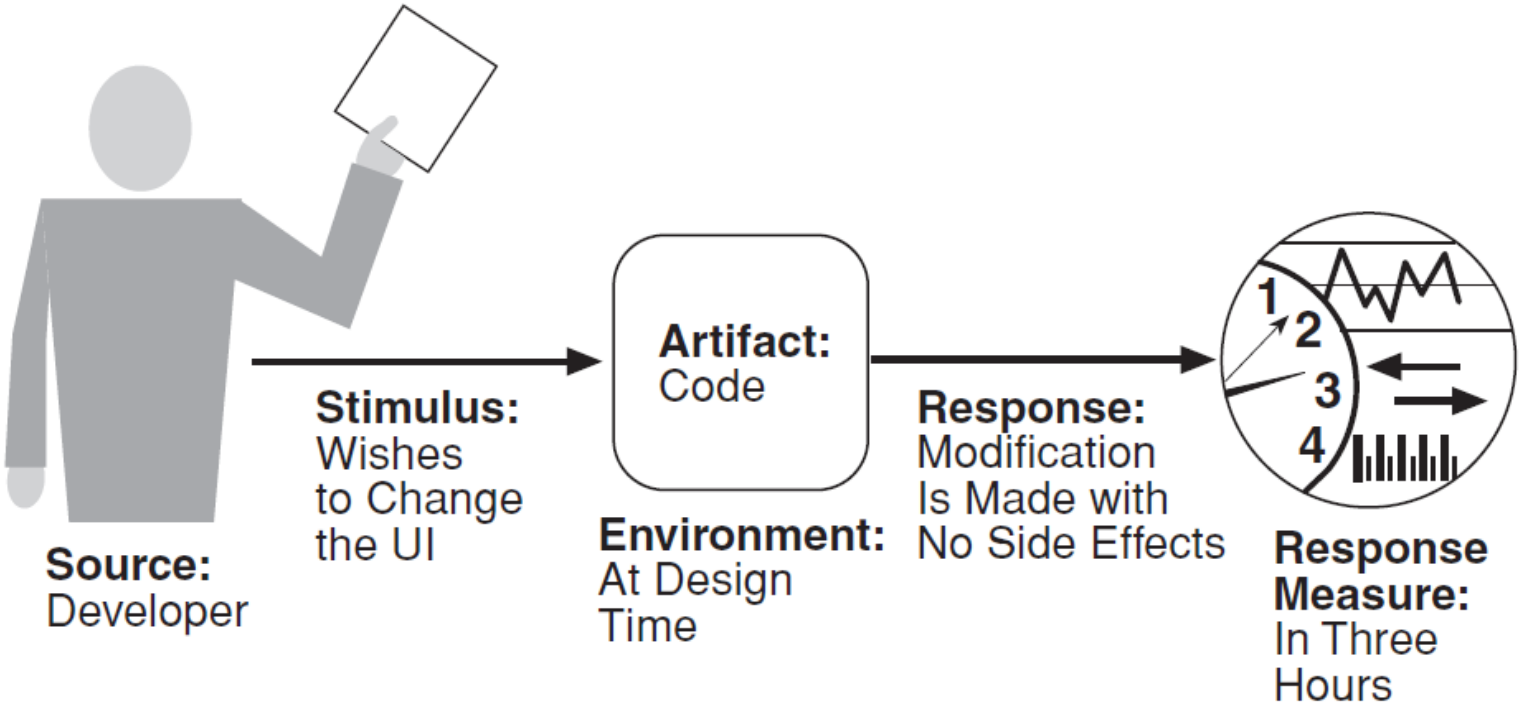
Justification for a change:

$$\mathbf{N \times C_n > C_i + N \times C_w}$$

Modifiability General Scenario

Portion of Scenario	Possible Values
Source	End user, developer, system administrator
Stimulus	A directive to add/delete/modify functionality, or change a quality attribute, capacity, or technology
Artifacts	Code, data, interfaces, components, resources, …
Environment	Runtime, compile time, build time, design time, …
Response	Make, test, or deploy modification
Response Measure	Cost in terms of the following: • Number, size, complexity of affected artifacts • Effort • Calendar time • Money (opportunity cost) • Extent to which this modification affects other functions or quality attributes • New defects introduced

A Concrete Scenario



Key Modifiability Ideas 1/2

Strike a balance between

- Cohesion:

  Mostly desired except coincidental and logical cohesion

- Coupling:

  Mostly undesired except data and message coupling

Key Modifiability Ideas 2/2

Allow late binding

An architecture that is equipped to accommodate modifications late in the life cycle, i.e. even after deployment

Tactics for Modifiability 1/8

Reduce size of a module

Split large modules into meaningful smaller modules

Tactics for Modifiability 2/8

Increase Cohesion

Increate semantic coherence within a module by moving excess responsibilities to other modules

Tactics for Modifiability 3/8

Restrict Dependencies

Restrict the visibility of internal modules so that they cannot be seen by developers

Tactics for Modifiability 4/8

Reduce Coupling - Encapsulate

Hide internal details and expose interface

Tactics for Modifiability 5/8

Reduce Coupling - Use an Intermediary

Don’t force data producers to know about the details of data consumers

Tactics for Modifiability 6/8

Reduce Coupling - Refactor

Detect and avoid code clones and large modules

Tactics for Modifiability 7/8

Reduce Coupling - Abstract Common Services

Group common features in abstract modules and implement variations in concrete modules

Tactics for Modifiability 8/8

Reduce Coupling - Defer Binding

Allow late changes in the life cycle, e.g. build script, makefile, compile-time parameterization, resource files, plugins, and so on

About the Design Process

Architectural design seldom starts from first principles

They are created as a process of selecting, tailoring, and combining patterns

Minimize upfront design - only design what is necessary (BDUF and YAGNI principles)

Architectural Patterns

Architectural patterns are design decisions found repeatedly in practice

They have know properties that permit reuse

A Design Problem

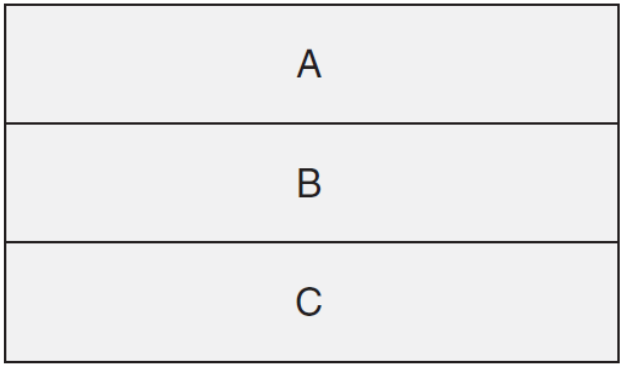
Modules need to be developed and evolved separately

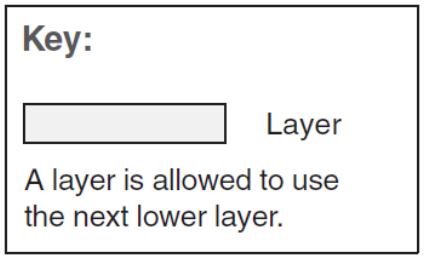
Modules may run on the same machine or different machines

Minimum amount of interaction among the modules

Must support portability, modifiability, and extensive reuse

Solution – Layered Pattern





Divide software into units called layer

Each layer is a grouping of modules that offers a cohesive set of services

Allowed-to-Use relationship among layers

Expose functionality of each layer through interfaces

Constraints

Every piece of software is allocated to exactly one layer

The allowed-to-use relations should not be circular

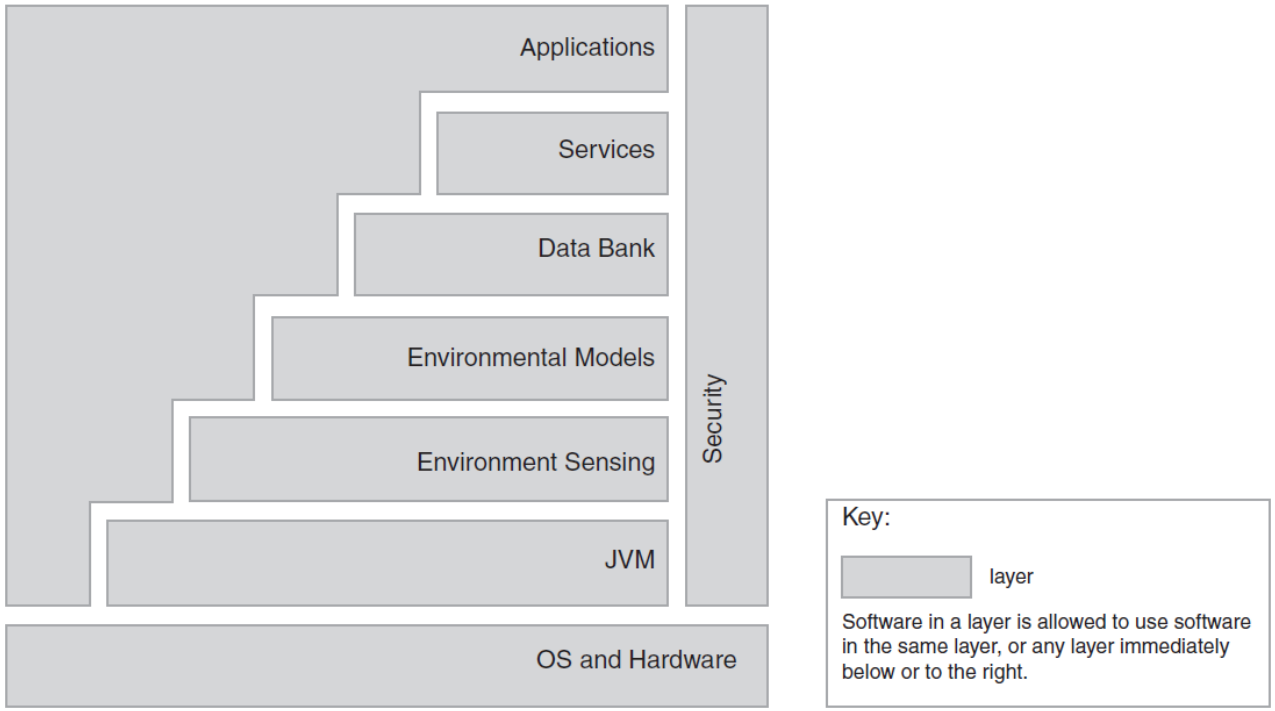
Constraints and Weaknesses

The addition of layers adds up-front cost and complexity to a system

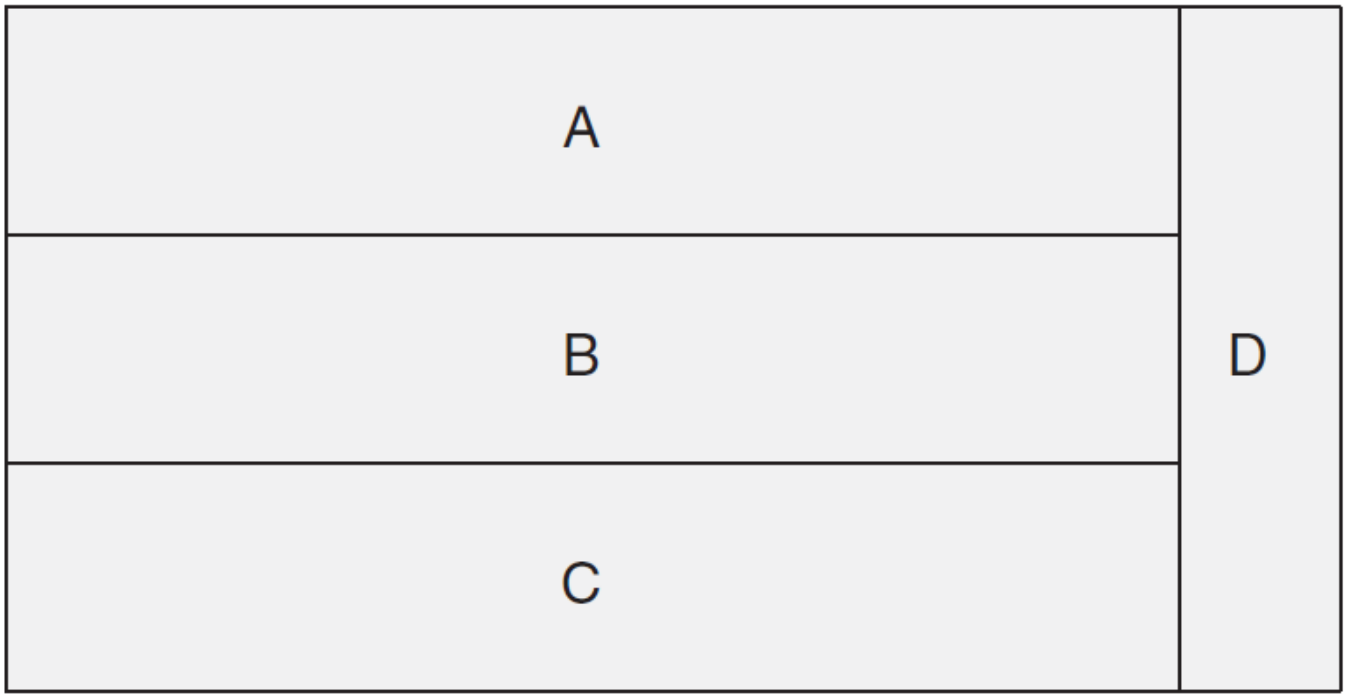
Layers contribute a performance penalty

Layer Bridging

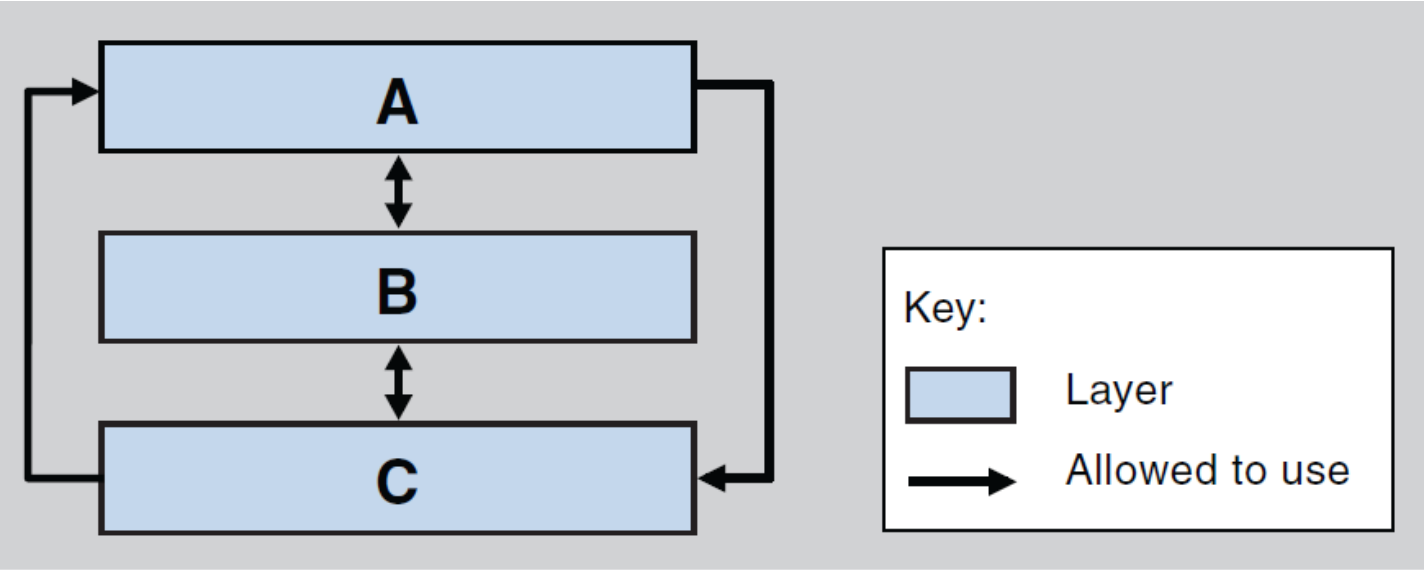
High-level layer may call non-adjacent lower-level layer



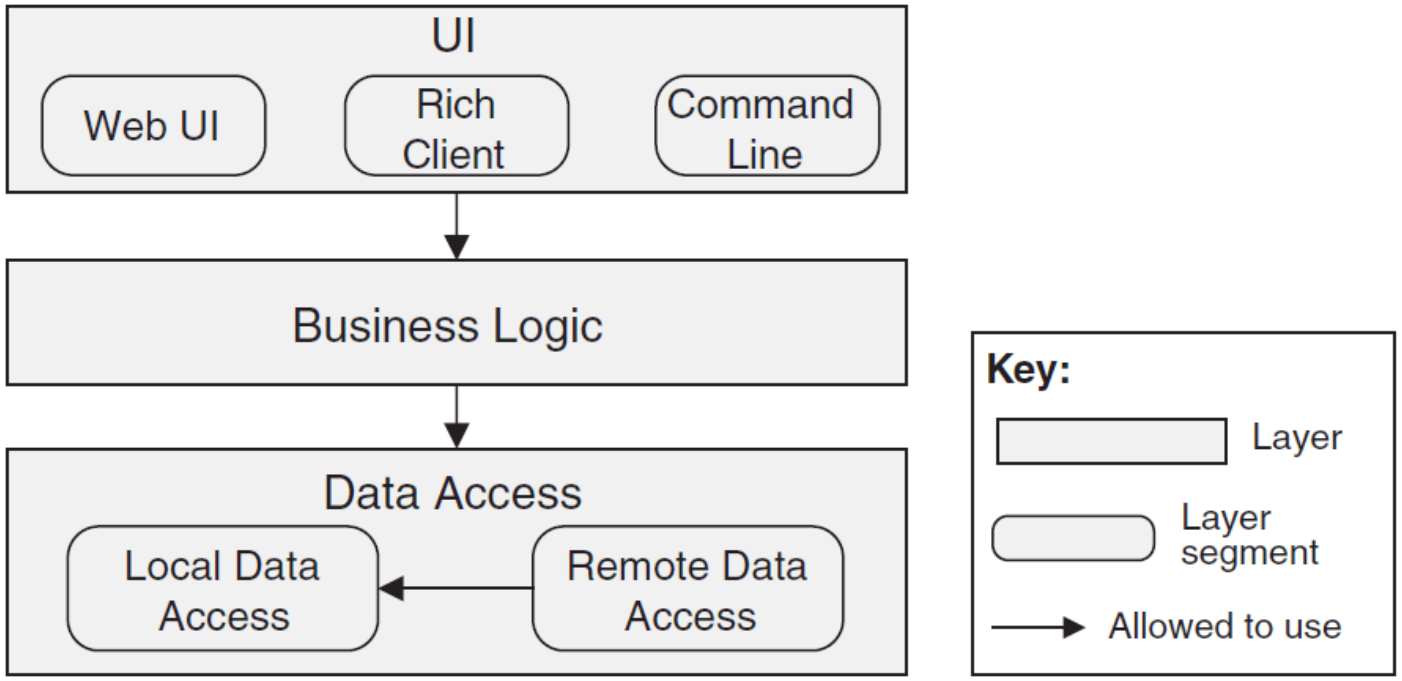
Is this Diagram a Layered Pattern?



How About this Diagram?



Segmented Layer



Next

Things Due

- **Quiz 1-2** due on Wednesday, 5:10 pm
- **Paper Review 1** due on Thursday, 8:00 am

Concepts

- Software Frameworks
- Pluggable Architecture