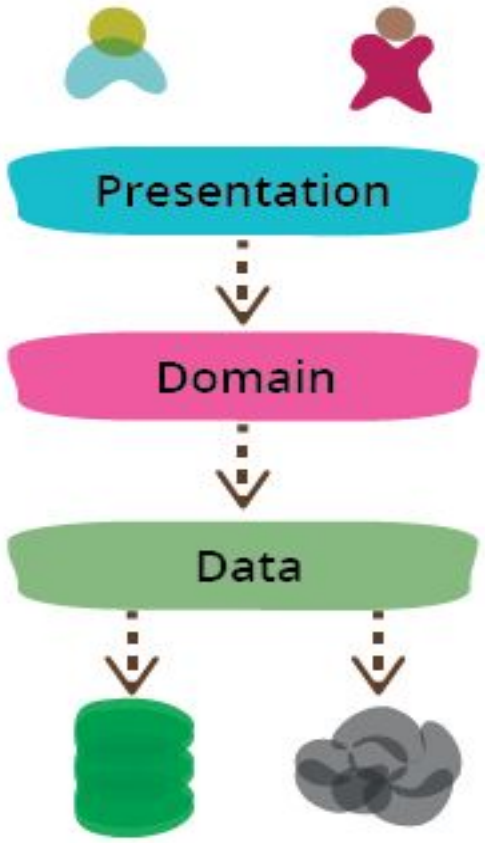
Layering Principles

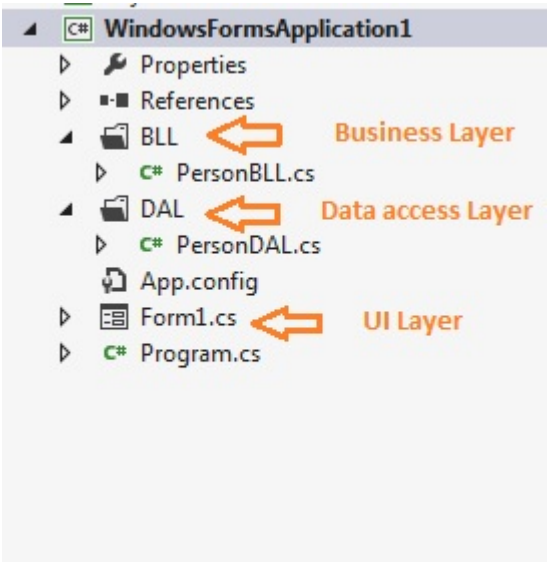
Software Design and Architecture (SWE 4601)

Based on https://martinfowler.com/bliki/LayeringPrinciples.html

Layered Architecture



- UI Layer User Interface related code
- Business Logic Layer business related calculation code
- Data Access Layer Database related code



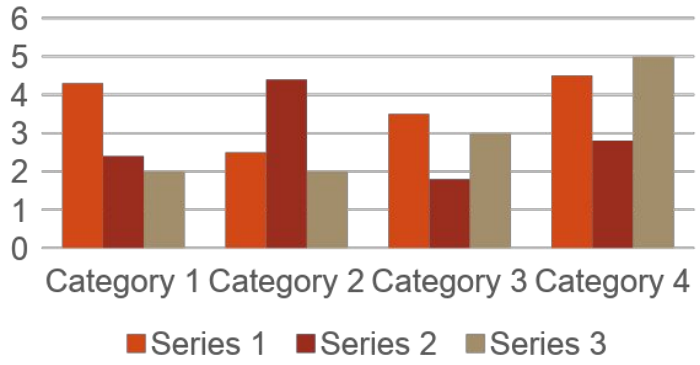
Layered Architecture Benefits

- Scope of Attention
  - Reducing
- Substitute different implementation
  - RDMS (shifting from MSSQL to Oracle)
  - You have built for web, now changing to mobile. You can replace/add another presentation layer
- Testing
  - Domain layer can be easily unit tested.
  - UI can be tested manually or automated way.

Disadvantage

- Create monolithic form of application
- Encourage layered specific team form
- Not appropriate for very small project because each layer add overhead

Layered Architecture



Req: Category wise sells data of December 2021

Presentation layer concerns

Domain Layer

Data Layer

[{“Toy”: “200”}, {“Dress”: “300”}]

[{“Toy”: “20%”}, {“Dress”: “30%”}]

Background

"For the last few days I've been attending a workshop on enterprise software in Norway, hosted by Jimmy Nilsson. During the workshop we had a session where we came up and ***voted on a bunch of design principles****.*

The rules were this. Everyone could propose principles for good layering which were added to a list. There was little discussion of the principle - only clarification as needed. People could propose principles they liked or not - the rule was to capture principles that people had heard in the field.

Once we'd got a list, it was time to vote. We used a variant of ***DotVoting*** *where everyone got 10 positive and 10 negative votes.*

I've put the results below - principle followed votes in the format positive/negative."

Martin Fowler, 7 January 2005

Results (1)

1. Low coupling between layers, high cohesion within them. 10/0
2. Separation of concerns. 11/0
3. Layers should be agnostic of consumers (a layer shouldn't know who's on top of it.) 4/4
4. User interface modules should contain no business logic. 10/0
5. Business logic layers contain no user interface and don't refer to user interface modules. 8/0
6. No circular references between layers. 8/0
7. There are at least three main layer types: presentation, domain, and data source. no 3/9
8. Business layer only uses abstractions of technological services. 14/0
9. Separate development teams by layer. no 1/22
10. Layers should be testable individual. 12/0
11. Prefer layers to interact only with adjacent layers. 4/4

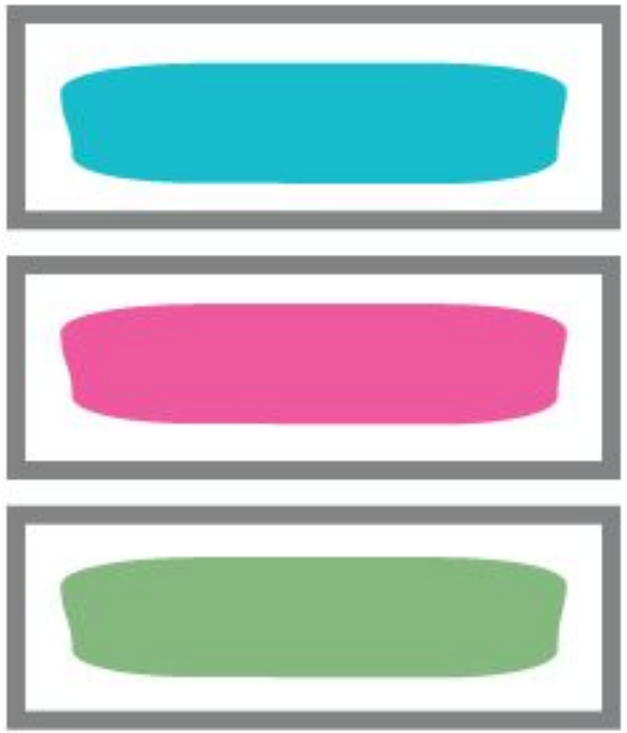
Results (2)

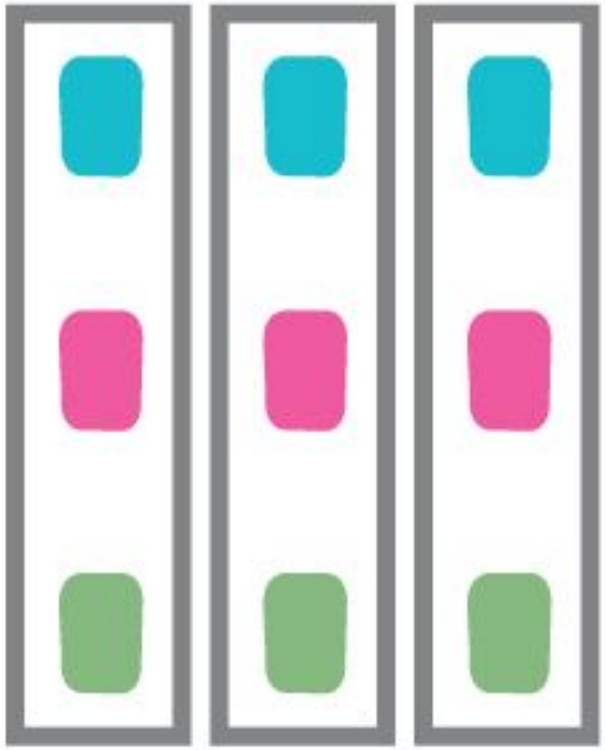
12. Layers should hide lower layers from upper layers.
13. Changing a lower level layer interface should not change upper layer interfaces. no 2/5
14. Distribute at layer boundaries no 0/18
15. Layers are a logical artifact that does not imply distribution between layers. 11/0
16. Lower layers should not depend on upper layers. 6/0
17. Layers should be substitutable. 2/0
18. Layers should be shy about their internals. 8/0
19. Always wrap domain logic with a service layer. 4/5
20. Rethrow exceptions at layer boundaries. no 0/15
21. Layers should have separate deployment units (e.g., separate jars or assemblies for each layer). 0/7

Results (3)

22. Layers may share infrastructural aspects (e.g., security) 7/0

23. Inbound external interface modules (e.g., web service handlers) should not contain business logic. 10/0





Don't use layers as the top level modules in a complex application...

... instead make your top level modules be full-stack

Application vs Enterprise vs System Architect

- The **Application Architect** is (or the Architect) The person responsible for the highest levels of design and scope for a particular solution/project.
  - You'd bother using *Application* in the title if there were other types of architects around, and you wanted it clear that this person worries mainly about a particular application.
- The **Enterprise Architect** is worried about *all* of a companies solutions. How they rely on each other, how they use each other, how efficient can their common upkeep and improvement be made.
  - He thinks about how all the solutions together support the company's mission. Only a larger company could warrant this grandiose title.
  - responsible for strategic thinking, roadmaps, principles, and governance of the entire enterprise. Usually has a close relationship with the business, vendors, and senior IT management.
  - The Enterprise Architect is a big shot who meets with the CIO, CTO, and other such big shots.

Application vs Enterprise vs System Architect

- The **Systems Architect** might be considered to have a wider scope than the Application Architect, and less than an Enterprise Architect.
  - This title is sometimes the exact same thing as Application Architect - big shot on a *particular* project. Person who duties include software but also *hardware* and *IT*, or someone worried about multiple applications.
  - Work within the framework laid down by the enterprise architecture team.