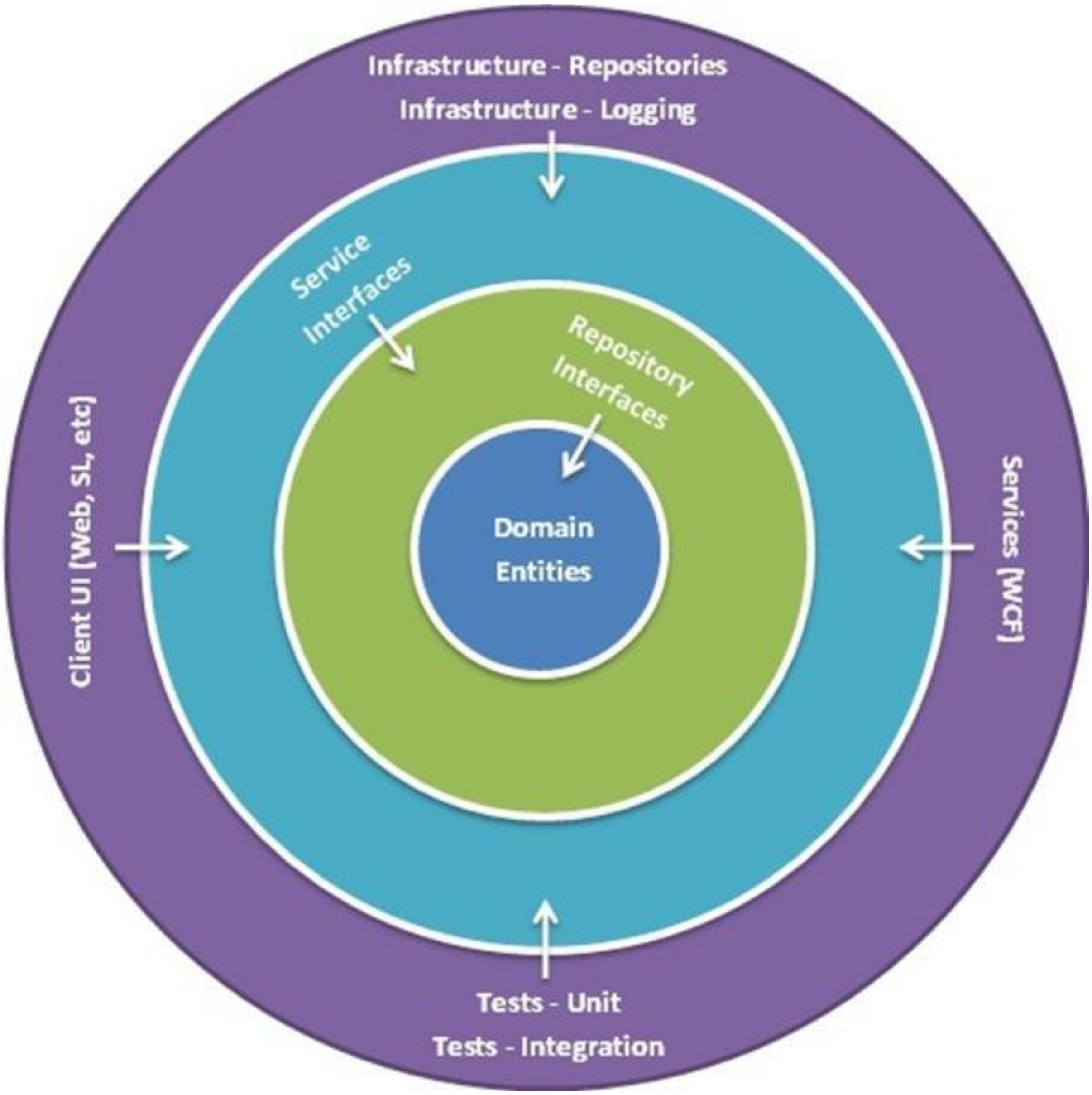
Onion Architecture

Desired Outcome

- Separation of concerns: Business logic separate from UI and Data logic
- Loose coupling: Emphasis is on dependency via interfaces
- Testability

The Architecture



In the Onion architecture, the outer layers depend on the inner layers, not the other way around. An inner layer must not depend on an outer layer.

At the core of the Onion is the object model, which represents the domain

Surrounding the object model are the repository and service inteface layers. Defining your repositories and services as interfaces decouples consumers (e.g. web controllers or unit tests) from concrete implementations. **Dependency Inject,** or Inversion of Control (IoC) is the key here. Let the consumers not worry about how or what implements the interfaces, and swapping implementations won't be an issue for the consumers.

The Infrastructure layer is the outermost layer. This represents Data access, UI, Looing, Tests, etc.

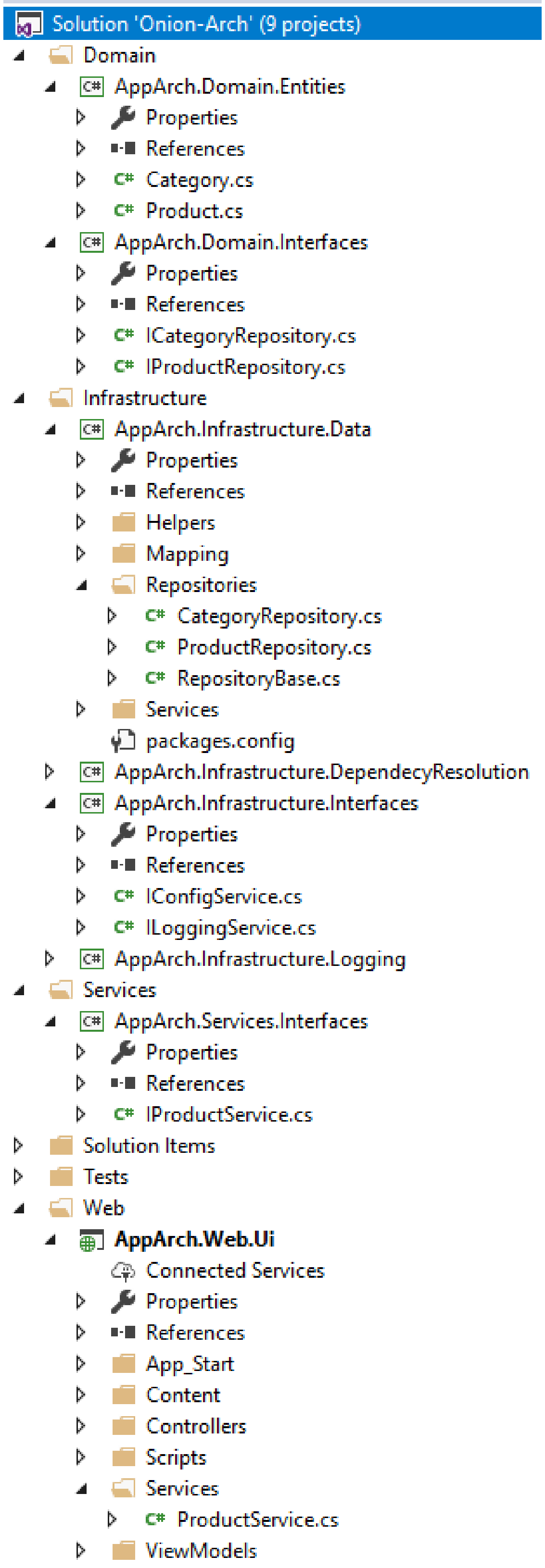
In Summary:

- The application is built around an independent object model
- Inner layers define interfaces. Outer layers implement interfaces
- Direction of coupling is toward the center
- All application core code can be compiled and run separate from infrastructure

How to organize application in visual studio

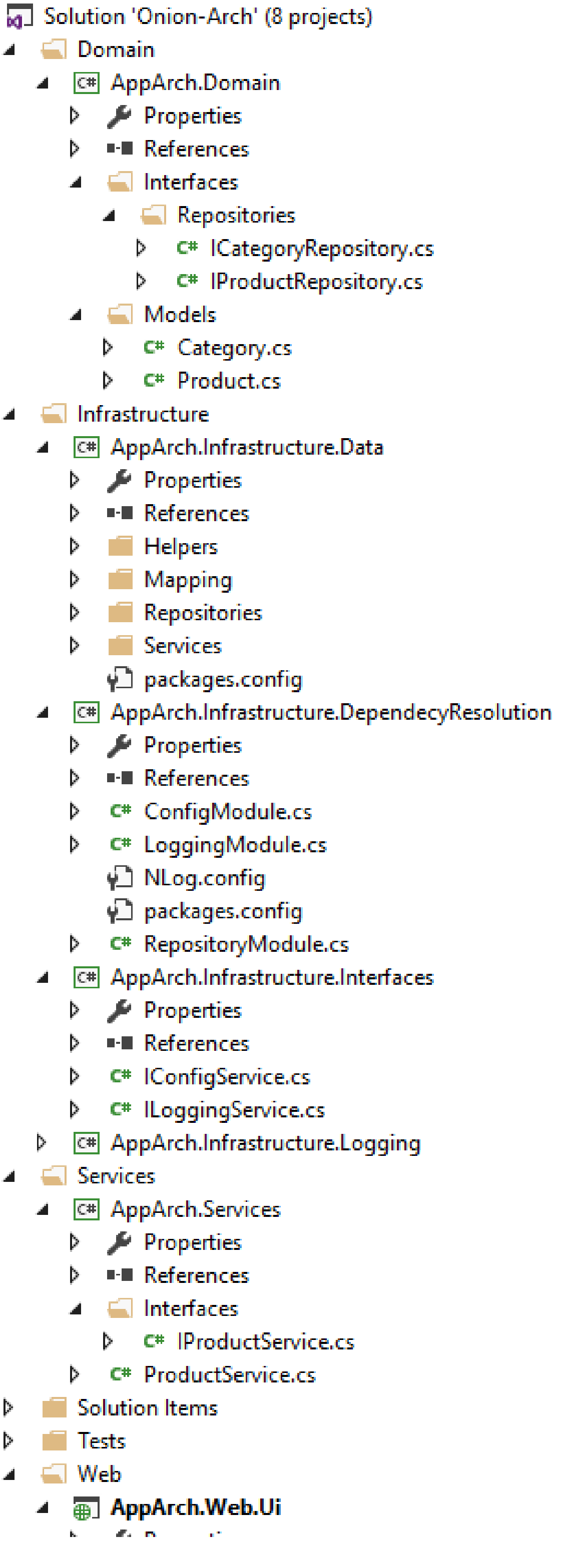
1. Domain
    - Entities/Models
    - Repository Interfaces
2. Infrastructure
    - Data accessors implementing repo interfaces and models in **Domain**
    - Other complimentary services such as logging
    - Interfaces for complimentary services such as logging
3. Services
    - Business logic consuming repositories in **Infrastructure** and models in **Domain**
    - May only contain interfaces for outer layer to consumer. Depends on app situation or dev preferences.
4. Application/UI
    - Consumes or implements interfaces in **Services**

Example project structure:



```
Solution 'Onion-Arch' (9 projects)
  Domain
    AppArch.Domain.Entities
      Properties
      References
      Category.cs
      Product.cs
    AppArch.Domain.Interfaces
      Properties
      References
      ICategoryRepository.cs
      IProductRepository.cs
  Infrastructure
    AppArch.Infrastructure.Data
      Properties
      References
      Helpers
      Mapping
      Repositories
        CategoryRepository.cs
        ProductRepository.cs
        RepositoryBase.cs
      Services
      packages.config
    AppArch.Infrastructure.DependecyResolution
    AppArch.Infrastructure.Interfaces
      Properties
      References
      IConfigService.cs
      ILoggingService.cs
    AppArch.Infrastructure.Logging
  Services
    AppArch.Services.Interfaces
      Properties
      References
      IProductService.cs
  Solution Items
  Tests
  Web
    AppArch.Web.Ui
      Connected Services
      Properties
      References
      App_Start
      Content
      Controllers
      Scripts
      Services
        ProductService.cs
      ViewModels
```

Simplified (Repo interfaces in Domain and Serivce logic not in UI/Web project)



```
Solution 'Onion-Arch' (8 projects)
  Domain
    AppArch.Domain
      Properties
      References
      Interfaces
        Repositories
          ICategoryRepository.cs
          IProductRepository.cs
      Models
        Category.cs
        Product.cs
  Infrastructure
    AppArch.Infrastructure.Data
      Properties
      References
      Helpers
      Mapping
      Repositories
      Services
      packages.config
    AppArch.Infrastructure.DependecyResolution
      Properties
      References
      ConfigModule.cs
      LoggingModule.cs
      NLog.config
      packages.config
      RepositoryModule.cs
    AppArch.Infrastructure.Interfaces
      Properties
      References
      IConfigService.cs
      ILoggingService.cs
    AppArch.Infrastructure.Logging
  Services
    AppArch.Services
      Properties
      References
      Interfaces
        IProductService.cs
      ProductService.cs
  Solution Items
  Tests
  Web
    AppArch.Web.Ui
```

Sources:

https://blog.tonysneed.com/2011/10/08/peeling-back-the-onion-architecture/

https://jeffreypalermo.com/2013/08/onion-architecture-part-4-after-four-years/