Problems the Onion architecture solves:
* Encourages loose coupling
* Makes unit testing easier
* Encapsulates business domain logic

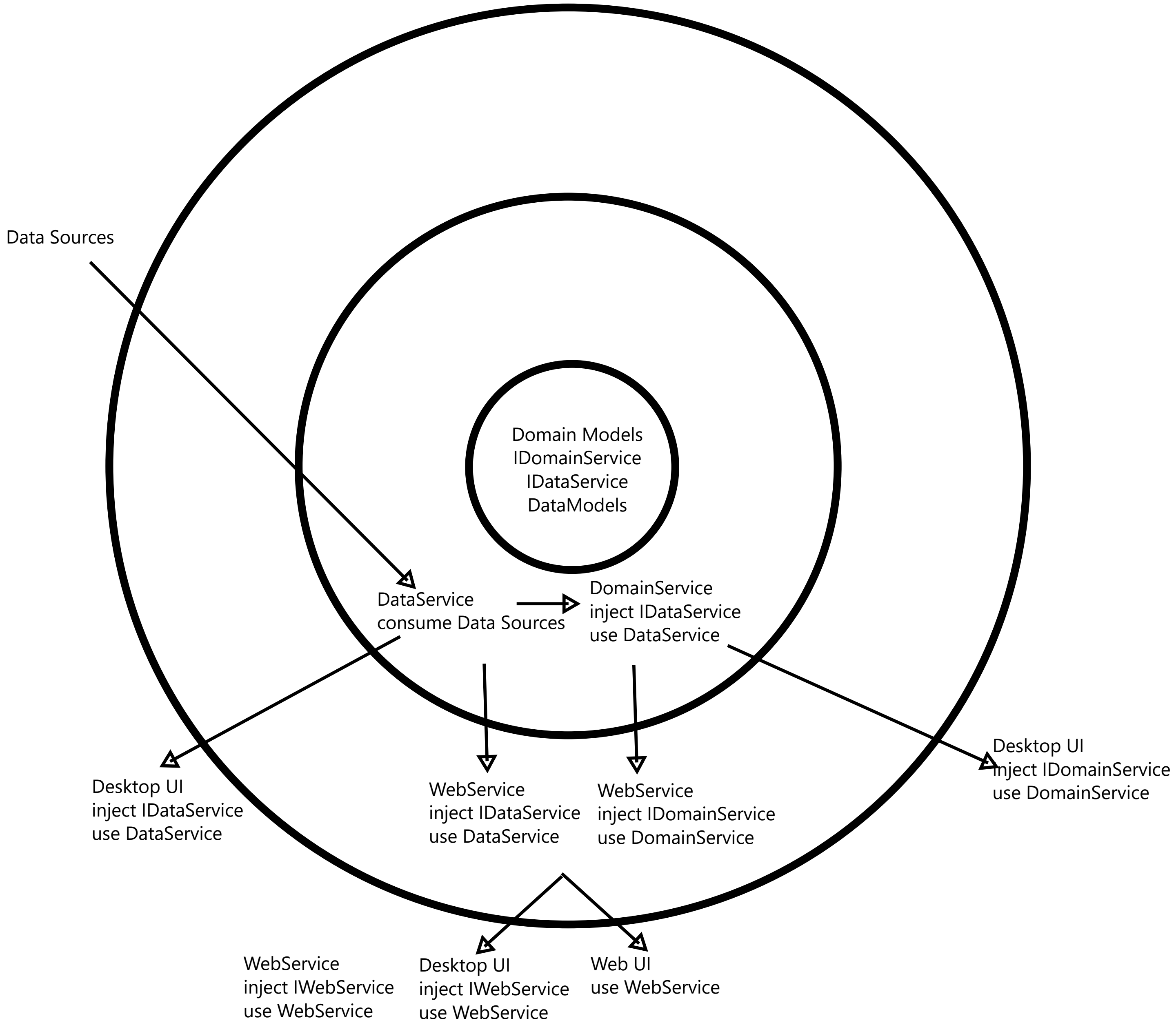
The key techniques involved are
* Abstract > Concrete (Interfaces)
* Dependency Injection
* Data Passing (DTOs)
* Data Mapping

* Which services make the most changes? These are where dependency injection matter most for unit testing.

  * Domain
  * UI

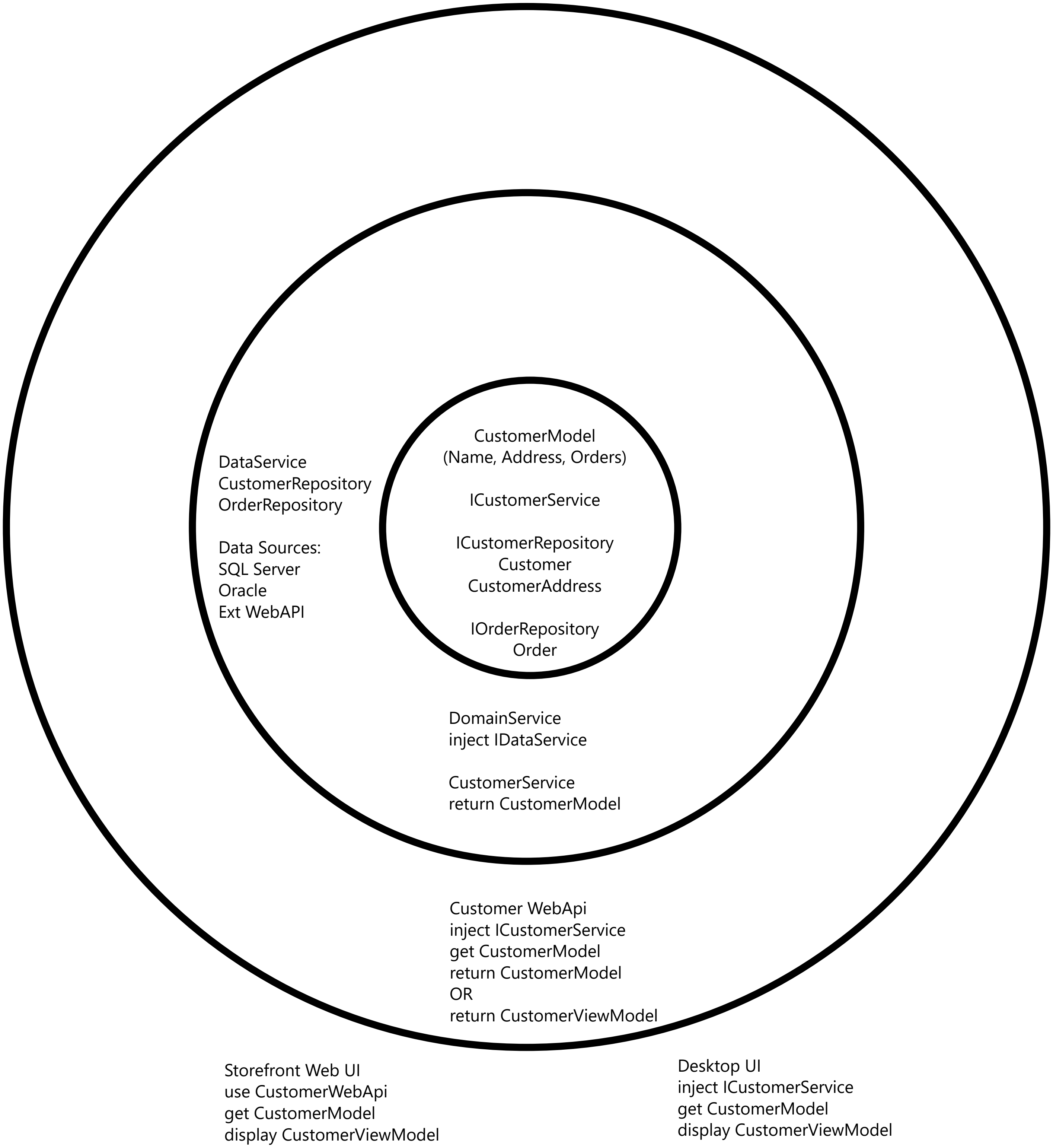
There are two kinds of models

* Domain (rich, could validate, business rules, etc)
* Data (simple, data transfer object)



Less flexible, easier to unit test

* Easy to mock in domain service
* Tempting to make data models match domain models
* Tempting to make just like DbContext



CustomerModel
(Name, Address, Orders)

ICustomerService

ICustomerRepository
Customer
CustomerAddress

IOrderRepository
Order

DataService
CustomerRepository
OrderRepository

Data Sources:
SQL Server
Oracle
Ext WebAPI

DomainService
inject IDataService

CustomerService
return CustomerModel

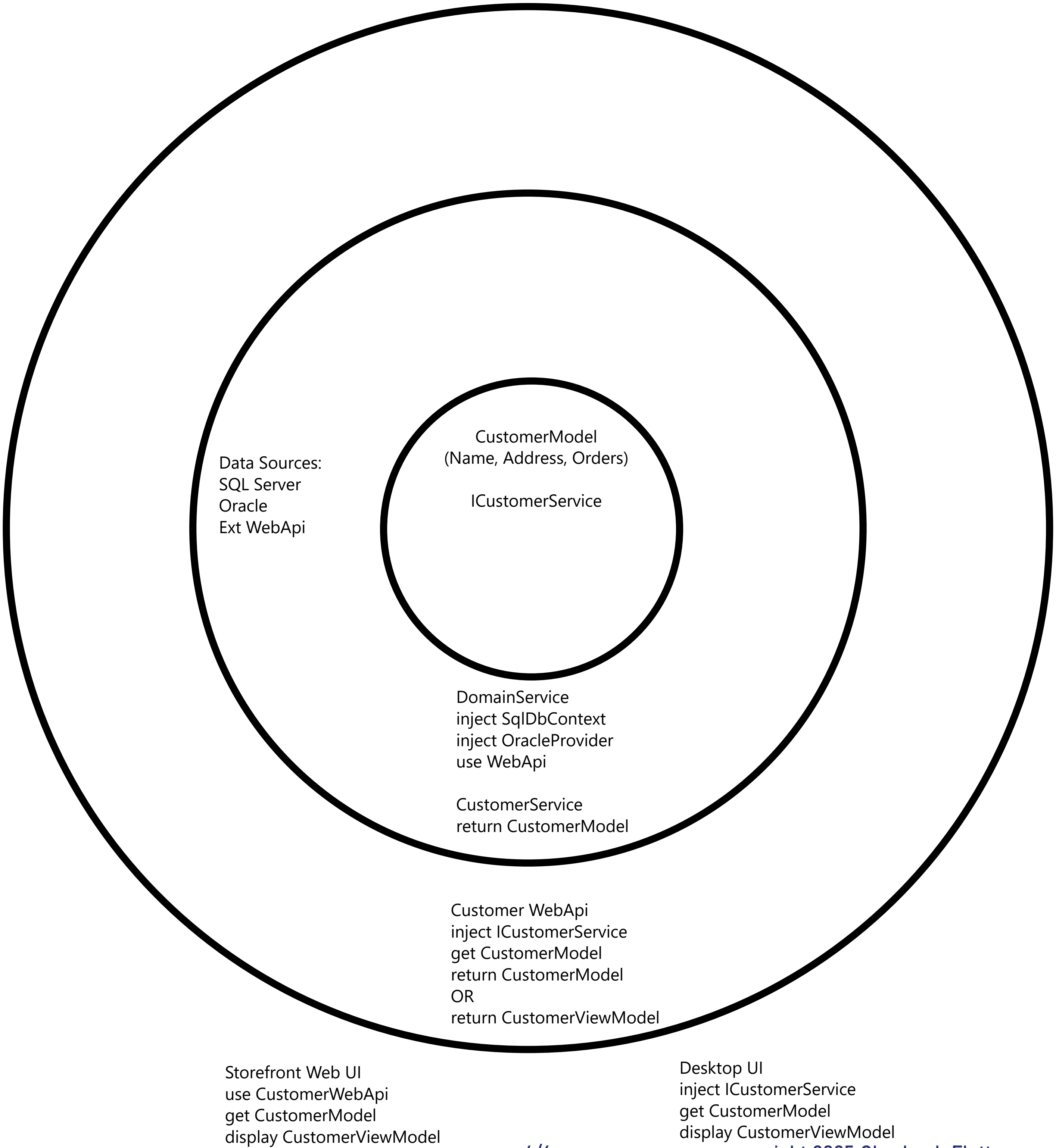
Customer WebApi
inject ICustomerService
get CustomerModel
return CustomerModel
OR
return CustomerViewModel

Storefront Web UI
use CustomerWebApi
get CustomerModel
display CustomerViewModel

Desktop UI
inject ICustomerService
get CustomerModel
display CustomerViewModel

More flexible, harder to unit test

* No mocking, inject in-memory or local database
* Unit tests worry about database state
* Tempting to over-seed database, or use real data



CustomerModel
(Name, Address, Orders)

ICustomerService

Data Sources:
SQL Server
Oracle
Ext WebApi

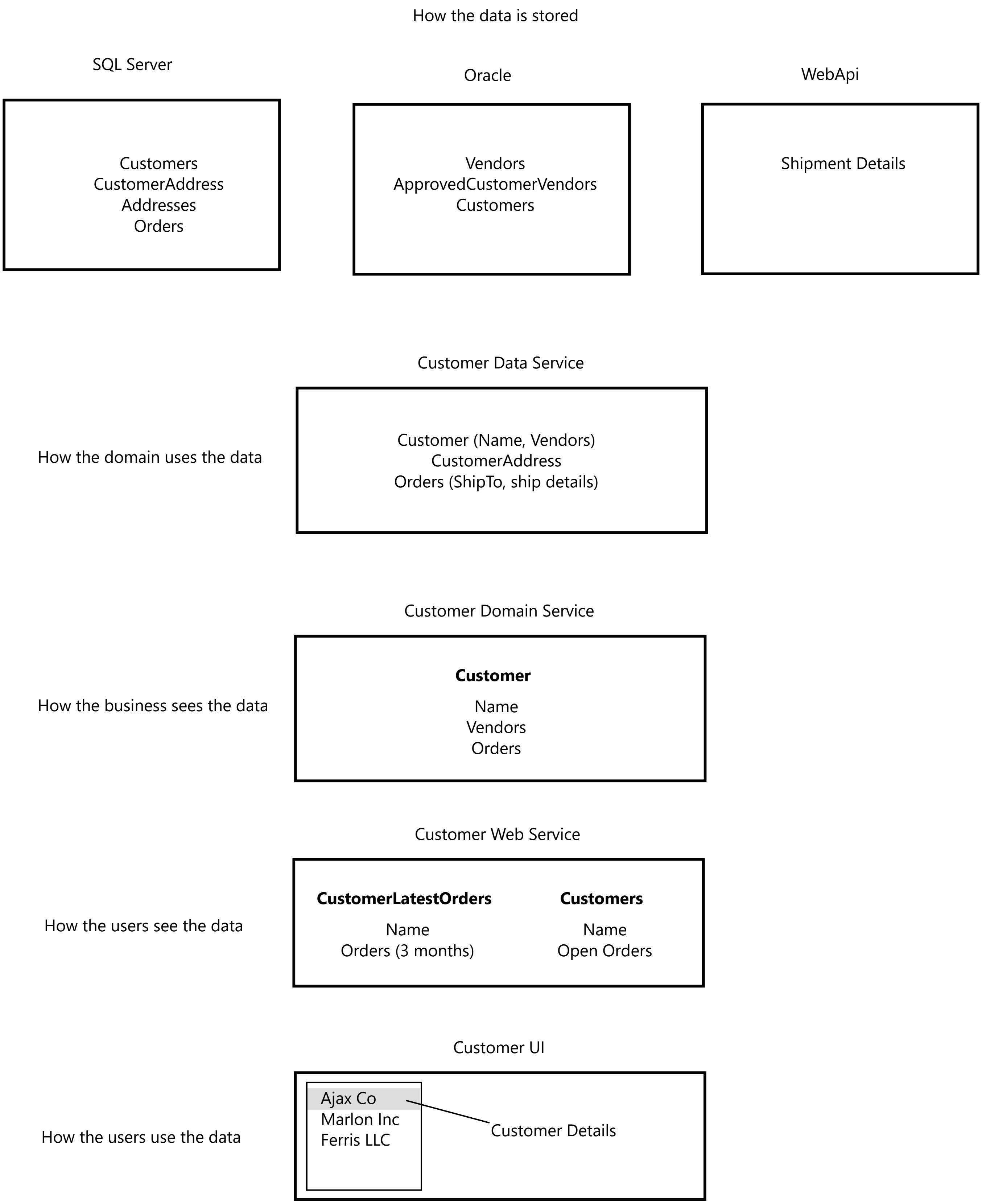
DomainService
inject SqlDbContext
inject OracleProvider
use WebApi

CustomerService
return CustomerModel

Customer WebApi
inject ICustomerService
get CustomerModel
return CustomerModel
OR
return CustomerViewModel

Storefront Web UI
use CustomerWebApi
get CustomerModel
display CustomerViewModel

Desktop UI
inject ICustomerService
get CustomerModel
display CustomerViewModel



How the data is stored

SQL Server

Customers
CustomerAddress
Addresses
Orders

Oracle

Vendors
ApprovedCustomerVendors
Customers

WebApi

Shipment Details

How the domain uses the data

Customer Data Service

Customer (Name, Vendors)
CustomerAddress
Orders (ShipTo, ship details)

How the business sees the data

Customer Domain Service

Customer

Name
Vendors
Orders

How the users see the data

Customer Web Service

CustomerLatestOrders

Name
Orders (3 months)

Customers

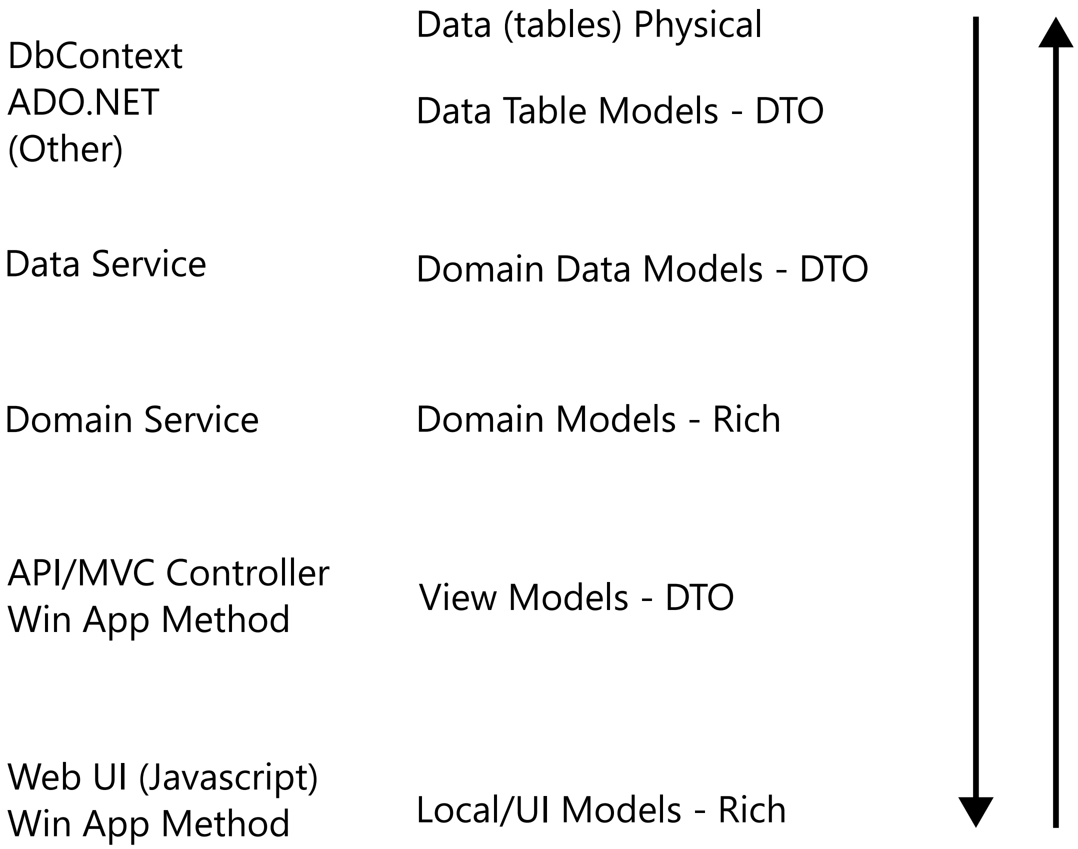
Name
Open Orders

How the users use the data

Customer UI

Ajax Co
Marlon Inc
Ferris LLC

Customer Details



	Data (tables) Physical
DbContext ADO.NET (Other)	Data Table Models - DTO
Data Service	Domain Data Models - DTO
Domain Service	Domain Models - Rich
API/MVC Controller Win App Method	View Models - DTO
Web UI (Javascript) Win App Method	Local/UI Models - Rich