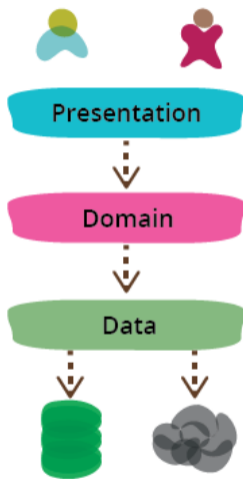


1 Class-Level Advice

- Monolithic Class
- Data Class

2 Architecture Advice

- Layering
- Protecting The Domain Layer



Why Separate Layers?

- Understandability, each layer has a coherent set of responsibilities, concerns are separated
- Substitutability, e.g. easy to substitute different front ends or data stores
- Testability
- Supports parts of the system changing at different rates
- Team specialisation

Which Layer's Code Is The Most Valuable?

- The domain layer **contains the business IP**
- Presentation and data access layers are typically built from, and use, standard components and frameworks

The domain layer is special \Rightarrow isolate and protect this layer

1 Class-Level Advice

- Monolithic Class
- Data Class

2 Architecture Advice

- Layering
- Protecting The Domain Layer

The domain layer should

- make use of a *ubiquitous language* (this part of the system should reflect the problem domain in a very literal way, so the mapping is obvious)
- contain a clean, expressive domain model **unpolluted by infrastructure concerns**
- have limited exposure to external components beyond the team's control — these represent risk if the components change (eg. Web API's)

“You don't want the domain model to [directly] depend on anything that talks to any kind of external system” ”

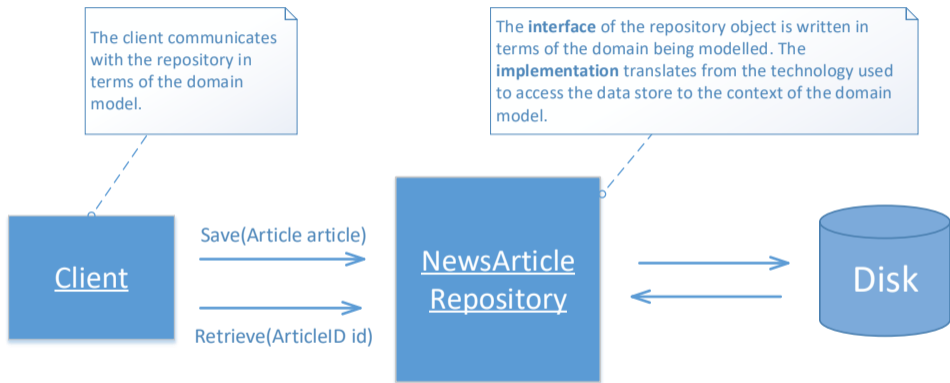
— Mathias Verraes

- Isolation from the presentation layer happens by default if the dependencies are right
- Isolation from the data store, and other services, requires you to write your own classes or interfaces which shield your domain from these external components (eg. the data access layer)

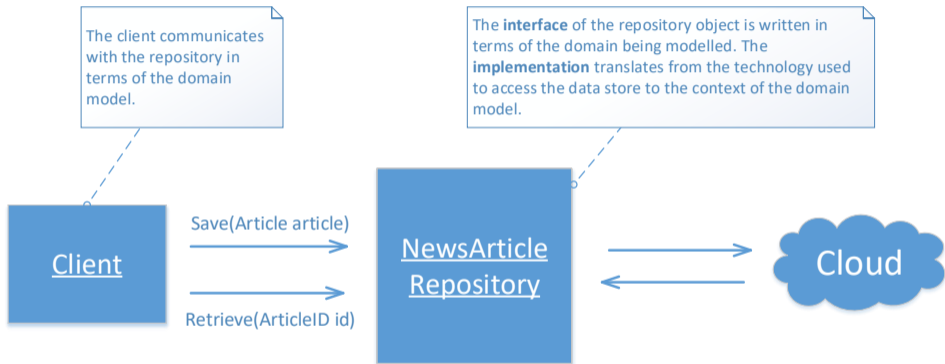
The screenshot shows the IOL News website with the following content:

- Navigation:** IOL logo, menu items (NEWS, SPORT, TECH, BUSINESS REPORT, ENTERTAINMENT, LIFESTYLE, MOTORING, TRAVEL, PERSONAL FINANCE), and a NEWSLETTER SUBSCRIBE button.
- Article Grid:**
 - Google starts to play nice with smaller rivals:** Software and Internet category, by Reuters, Oct 3, 2020. Image shows the Google logo.
 - Roblox gets ready for stock market launch:** Gaming category, by Reuters, Oct 3, 2020. Image shows Roblox avatars.
 - SA Innovation Summit 2020 ends on positive and successful note:** Software and Internet category, by Floyd Matlala, Oct 2, 2020. Image shows a Zoom meeting titled "SAIS VIRTUAL 365 CLOSING CEREMONY".
 - SA Innovation Summit 2020: These are the Africa Cup winners:** Software and Internet category, by Faheem Khota, Oct 2, 2020. Image shows product packaging.
- Editors Choice / Most Read:**
 - Google starts to play nice with smaller rivals
 - 'Creaks' game review: Why you should play it even though it might melt your brain
 - Roblox gets ready for stock market launch
 - WhatsApp update will help free up space on your phone
 - SA law firm takes on Facebook in court battle and wins
- Trending Section:** TRENDING ON IOL with hashtags #LOCALISLEKKER, #SAIS2020, #CORONAVIRUS, and #STATECAPTUREINQUIRY.

Isolation From The Data Store



Isolation From the Data Store cont.



Using Third-Party Libraries Within The Domain Layer

- Avoid re-inventing the wheel, use third-party libraries and classes which are well-written, stable, tested, and maintained
- Consequences
 - Tightly coupling the domain logic to code you do not own/control
 - **Be wary** of frameworks and libraries that force you to compromise your design
- Use library classes *as is* if they represent genuinely useful abstractions
 - `boost::scoped_ptr` was the forerunner of `unique_ptr`
- If necessary, *wrap* library classes (using composition) to make them meaningful for the domain, and to expose only the methods that the domain requires
 - In a “shipping and delivery” domain create a `DeliveryDate` class which internally uses `boost::date_time` in order to exclude delivery dates on weekends