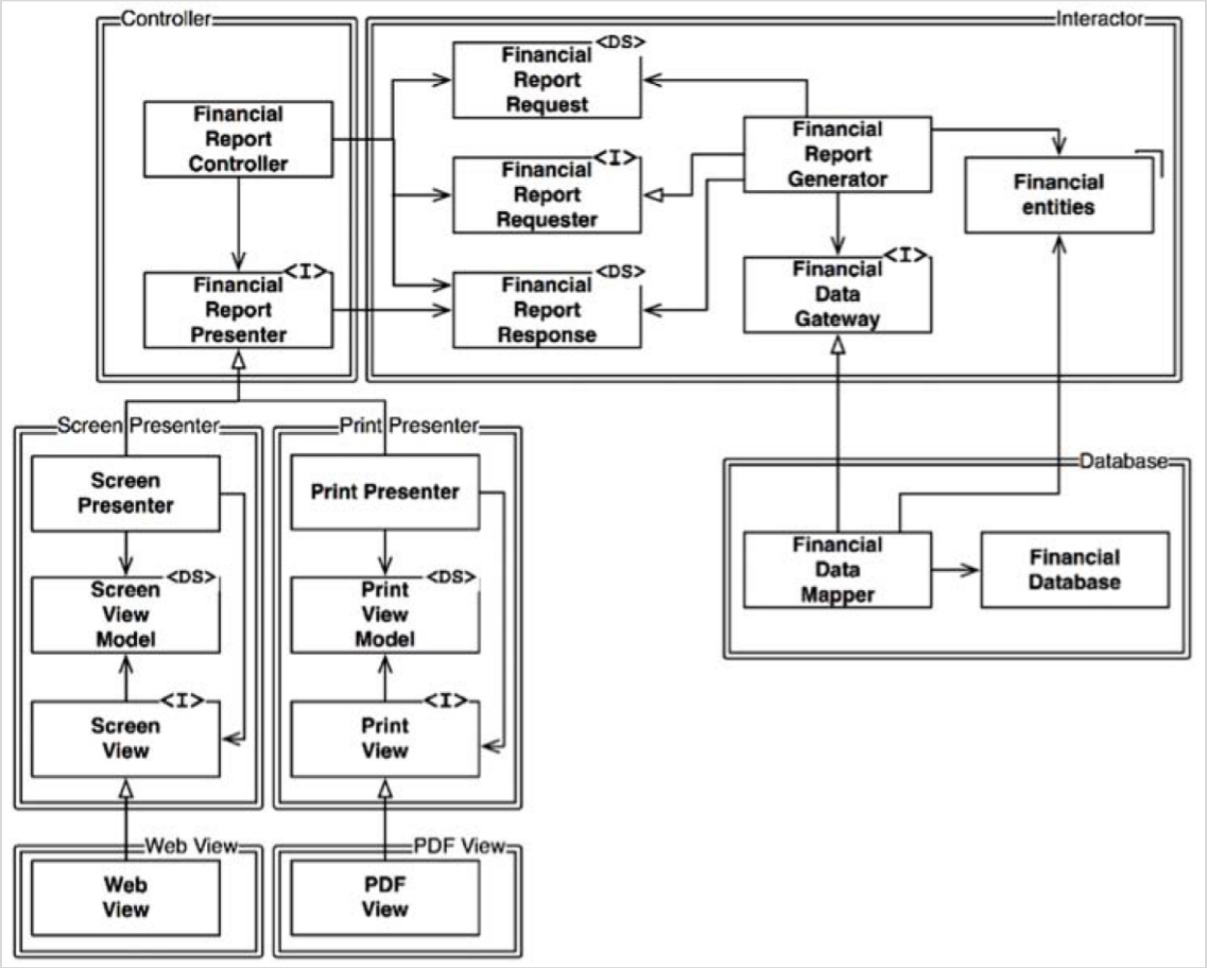
Chapter 8: The Open-Closed Principle

> A software artifact should be open for extension but closed for modification



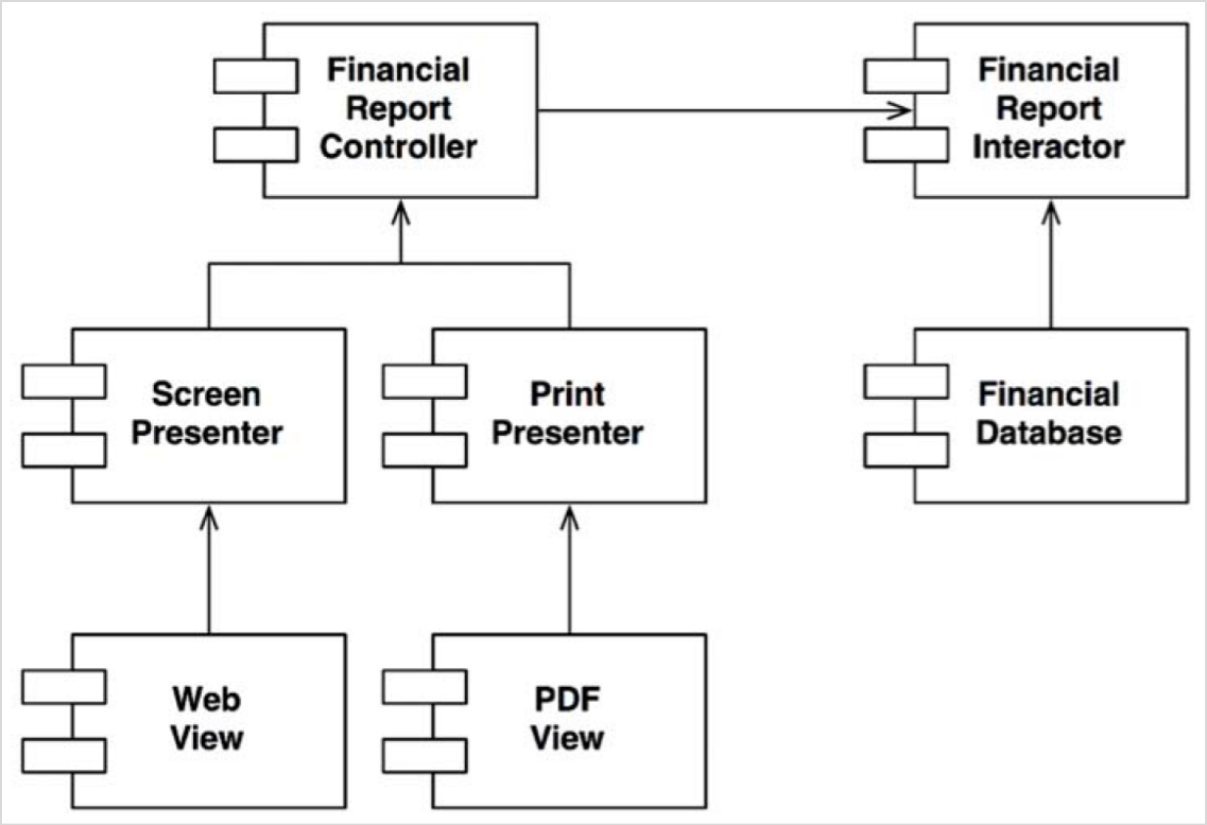
<I>: interfaces

<DS>: data structures

Open arrowheads: using relationships

Closed arrowheads: implements or inheritance relationships

- partition processes into classes, separating classes into components
- all dependencies are source code dependencies
  - FinancialDataMapper ➜ FinancialDataGateway
    - source code of FinancialDataMapper mentions FinancialDataGateway
    - FinancialDataGateway mentions nothing about FinancialDataMapper
- Double line is crossed in one direction only
  - All component relationships are unidirectional



- arrows point toward the components that we want to protect from change
- If component A should be protected from changes in component B, then component B should depend on component A
- the Interactor best conforms to the OCP
  - Changes to anything else will not impact it
  - it contains the highest-level policies of the app
- Views, one of the lowest-level concepts, are least protected
- (Lowest) Views ➜ Presenter ➜ Controller ➜ Interactor (Highest)
- Higher level components are protected from changes made to lower-level components

Directional Control

- FinancialDataGateway interface between the FinancialReportGenerator and the FinancialDataMapper exists to invert the dependency that would otherwise have pointed from the Interactor component to the Database component

Information Hiding

- FinancialReportRequester protects FinancialReportController from knowing too much about the internals of the Interactor
- If it weren't there, the Controller would have transitive dependencies on the FinancialEntities
  - Transitive dependency: software entities should not depend on things they don't

directly use

#pivotal/book-club/clean-architecture/chapter8