

Module 2:

"SOLID in Practice"

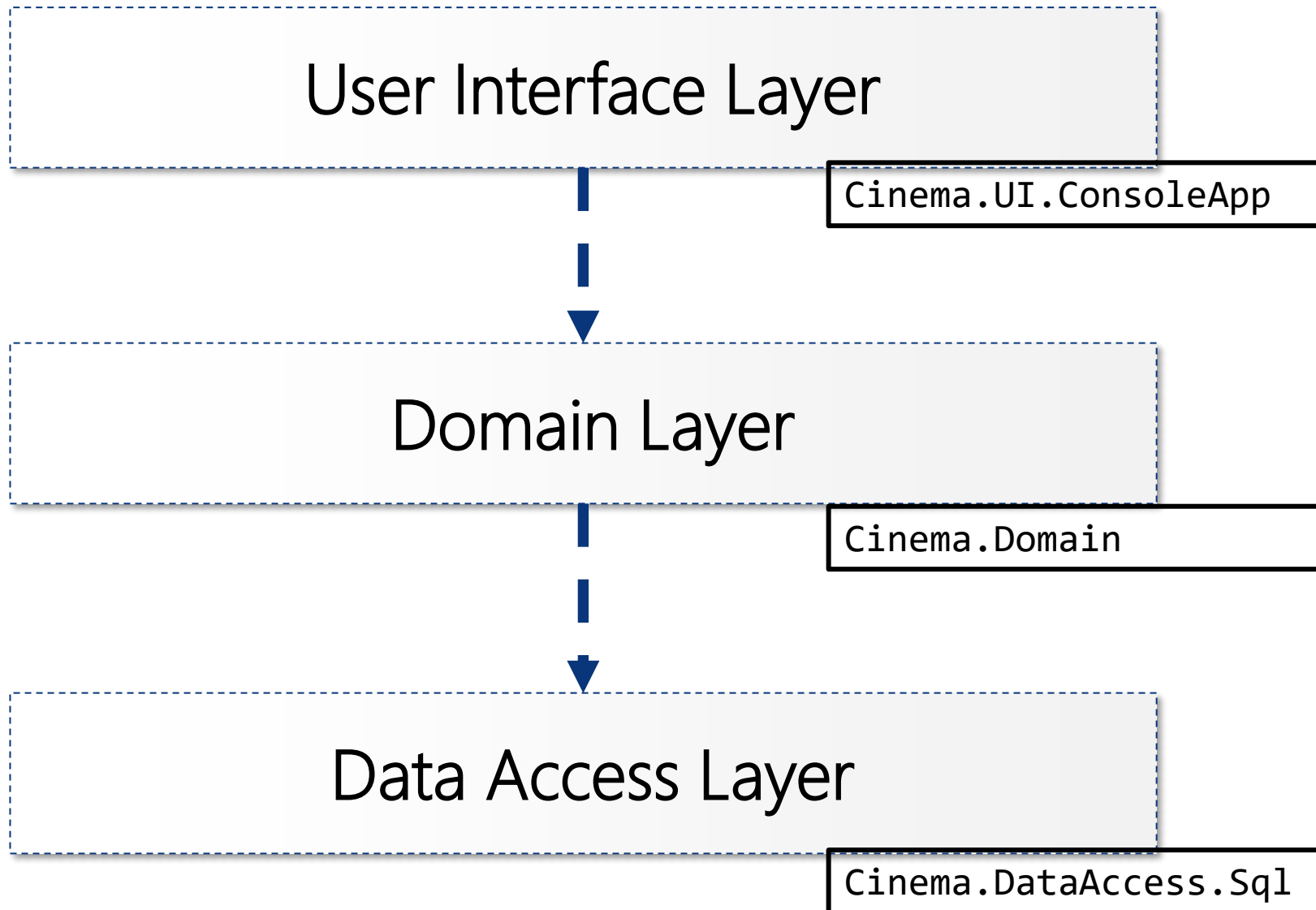


TEKNOLOGISK
INSTITUT

Agenda

- ▶ **Discussion: Evaluating the Design**
- ▶ *Pattern: Repository (with Entity Framework)*
- ▶ Workshop A.2: Data Access Layer with Repository
- ▶ Discussion: Evaluating the Design Again
- ▶ *(Optional) Automatic Testing*
- ▶ Workshop A.3: Test Domain and Change Data Access

Beautiful Layered Design?



Discussion:

Evaluating the Design

- ▶ Can we change the UI Layer from Console to e.g. Web or WPF?
- ▶ Can we unit test the Domain Layer?
- ▶ Can we change the Data Access Layer?

Anti-Pattern: Entourage

- ▶ *When A depends upon B, and you group B and C in the same assembly, then if C depends upon D, in effect, you have equipped A with a dependency upon D.*
- ▶ Outline
 - If you keep the interfaces and implementations in the same assembly, you essentially inherit dependencies' dependencies.
 - Entourage means ask for one assembly and it gives all its assemblies.
 - Nuget packages are potentially evil!
- ▶ See:
"Adaptive Code" (2nd Edition)
Gary McLean Hall (2017)

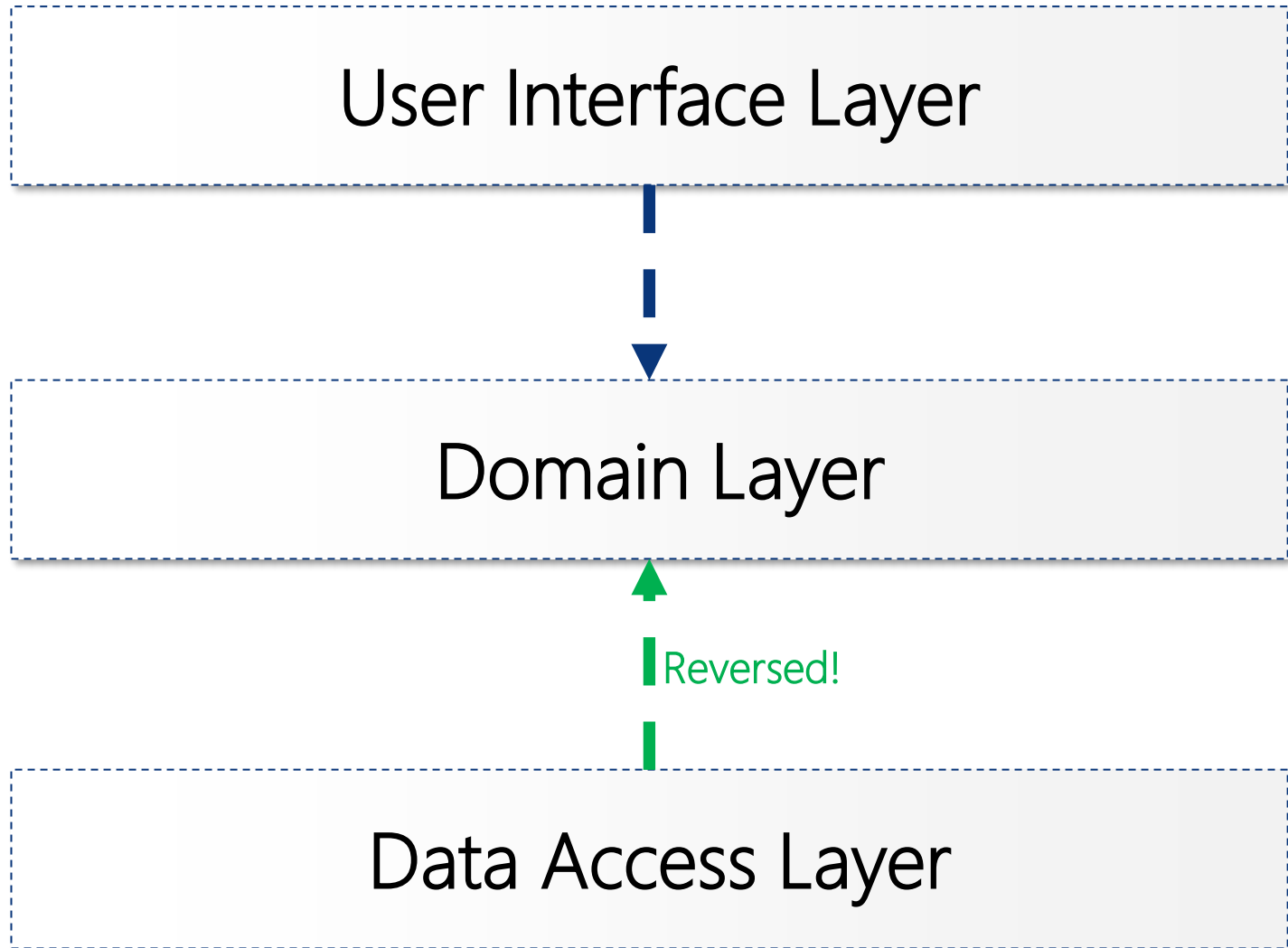
Agenda

- ▶ Discussion: Evaluating the Design
- ▶ ***Pattern: Repository (with Entity Framework)***
- ▶ Workshop A.2: Data Access Layer with Repository
- ▶ Discussion: Evaluating the Design Again
- ▶ *(Optional) Automatic Testing*
- ▶ Workshop A.3: Test Domain and Change Data Access

Agenda

- ▶ Discussion: Evaluating the Design
- ▶ *Pattern: Repository (with Entity Framework)*
- ▶ **Workshop A.2: Data Access Layer with Repository**
- ▶ Discussion: Evaluating the Design Again
- ▶ *(Optional) Automatic Testing*
- ▶ Workshop A.3: Test Domain and Change Data Access

Better SOLID Design





Workshop A.2: Data Access Layer with Repository



Agenda

- ▶ Discussion: Evaluating the Design
- ▶ *Pattern: Repository (with Entity Framework)*
- ▶ Workshop A.2: Data Access Layer with Repository
- ▶ **Discussion: Evaluating the Design Again**
- ▶ *(Optional) Automatic Testing*
- ▶ Workshop A.3: Test Domain and Change Data Access

Discussion:

Evaluating the Design

- ▶ Can we change the UI Layer from Console to e.g. Web or WPF?
- ▶ Can we unit test the Domain Layer?
- ▶ Can we change the Data Access Layer?

Pattern: Stairway

- ▶ *Let your implementation packages depend upon packages that exclusively contain interfaces (or interface-like classes). Moreover, your packages should not depend other implementation packages.*
- ▶ Outline
 - This is essentially the “module” part of DIP
 - Avoids the Entourage anti-pattern
 - May not always be practically manageable
- ▶ See:
“Adaptive Code” (2nd Edition)
Gary McLean Hall (2017)

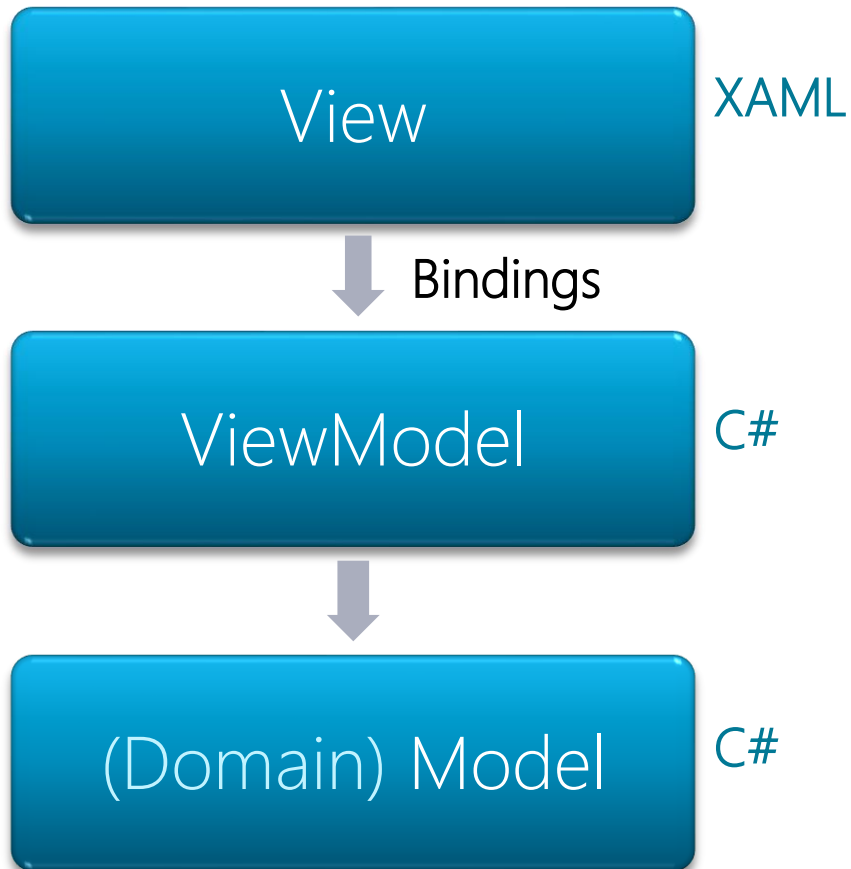
Implications of Stairway

- ▶ Keep the interfaces and implementations in the different assemblies
 - Can vary the two independently
 - clients only need to make a single reference—to the interface assembly.
- ▶ Interfaces should not have any external dependencies
 - As far as possible, this should always be adhered to
- ▶ Interfaces should not have methods or properties that expose any data objects or classes defined in third-party references
 - A reference to infrastructural entities (i.e. third-party dependencies) should be avoided.

Unfortunately...

- ▶ Third party library such as Log4Net, NHibernate, and MongoDB are packaged using the **Entourage anti-pattern**.
- ▶ **Solution:**
To work around the above issue, make use of a simple interface that hides the third-party dependency behind a first-party dependency and an adapter

Pattern: Model-View-ViewModel

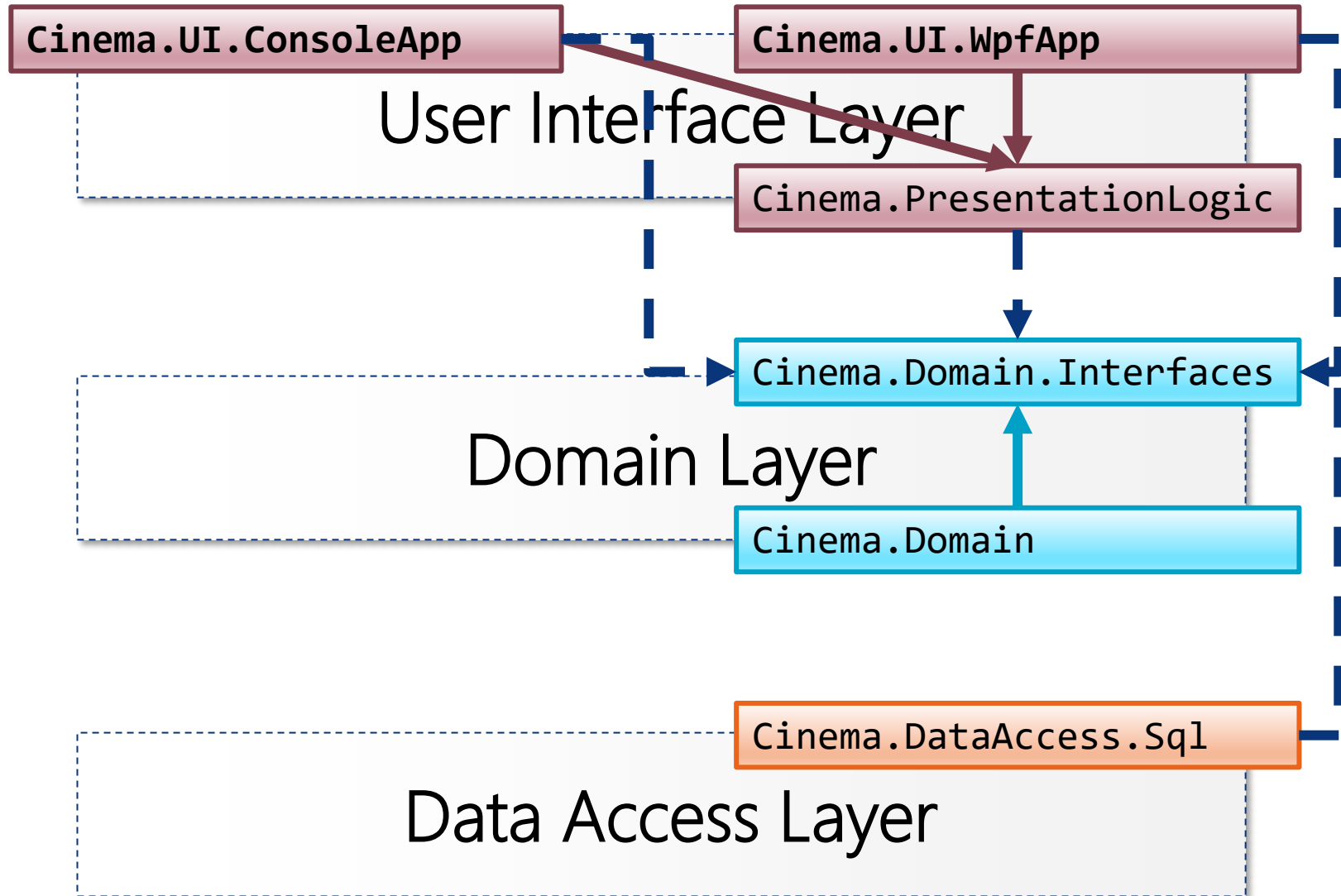


- ▶ Separation between presentation and application logic

- ▶ The ViewModel is an abstraction of the View

- ▶ Depends heavily on **data binding** and **command binding**

Stairway Design



Architectural Principles in Summary

- ▶ Ports and Adapters a.k.a. Hexagonal Architecture
 - Alistair Cockburn (2005)
 - <https://alistaircockburn.us/hexagonal-architecture/>

- ▶ Onion Architecture
 - Jeffrey Palermo (2008)
 - <https://jeffreypalermo.com/2008/07/the-onion-architecture-part-1/>

- ▶ Clean Architecture a.k.a. Screaming Architecture
 - Robert C. Martin (2012)
 - <https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html>

Agenda

- ▶ Discussion: Evaluating the Design
- ▶ *Pattern: Repository (with Entity Framework)*
- ▶ Workshop A.2: Data Access Layer with Repository
- ▶ Discussion: Evaluating the Design Again
- ▶ ***(Optional) Automatic Testing***
- ▶ Workshop A.3: Change the Data Access Layer

Agenda

- ▶ Discussion: Evaluating the Design
- ▶ *Pattern: Repository (with Entity Framework)*
- ▶ Workshop A.2: Data Access Layer with Repository
- ▶ Discussion: Evaluating the Design Again
- ▶ *(Optional) Automatic Testing*
- ▶ **Workshop A.3: Test Domain and Change Data Access**



Workshop A.3: Change the Data Access Layer



Summary

- ▶ Discussion: Evaluating the Design
- ▶ *Pattern: Repository (with Entity Framework)*
- ▶ Workshop A.2: Data Access Layer with Repository
- ▶ Discussion: Evaluating the Design Again
- ▶ *(Optional) Automatic Testing*
- ▶ Workshop A.3: Test Domain and Change Data Access



WINCUBATE

Jesper Gulmann Henriksen

PhD, MCT, MCSD, MCPD

Phone : +45 22 12 36 31

Email : jgh@wincubate.net

WWW : <http://www.wincubate.net>

Ringgårdsvej 4A

8270 Højbjerg

Denmark