

Clean Architecture

Andreas Richter



ar@anrichter.net



[@anrichter](https://twitter.com/anrichter)



www.anrichter.net

Agenda

Warum überhaupt
Software Architektur?

Was ist
Clean Architecture

Demo

Zusammenfassung &
Quellen

Warum überhaupt Software Architektur?

Was ist eigentlich Software Architektur?

Definition nach IEEE-1471 – ISO 42010

“Die grundsätzliche Organisation eines Systems, wie sie sich in dessen **Komponenten**, deren **Beziehung** zueinander und zur Umgebung widerspiegelt, sowie die **Prinzipien**, die für einen Entwurf und seine Evolution gelten.” (Quelle: innoQ Deutschland)

Weitere Definitionen

<http://sei.cmu.edu/architecture/start/glossary/index.cfm>

Einflussfaktoren beim Softwareentwurf

Funktionale Anforderungen

(UseCases, Specs, ...)

Technische

(Hard- & Softwareinfrastruktur, OS, Development Tools, ...)

Organisatorische

(Zeit, Budget, Team, Vorgehensmodell, ...)

Nichtfunktionale Anforderungen

Nichtfunktionale Anforderungen

Auch Qualitätsanforderungen genannt (arc42)

Ermöglichen Aussage über gute oder schlechte Architektur

Essentiell für das Entwerfen einer Architektur

Ohne droht ein „Struktureller Monolith“

Wartbarkeit

Portierbarkeit

Testbarkeit

...

Clean Architecture



SOLID Prinzipien

Single Responsibility Principle

"Es sollte nie mehr als einen Grund geben, eine Klasse zu ändern"

Open-Closed Principle

"Module sollten sowohl offen (für Erweiterungen), als auch geschlossen (für Modifikationen) sein"

Liskovsche Substitution Principle

"Eigenschaften einer Basisklasse sollten in Subklassen erhalten bleiben"

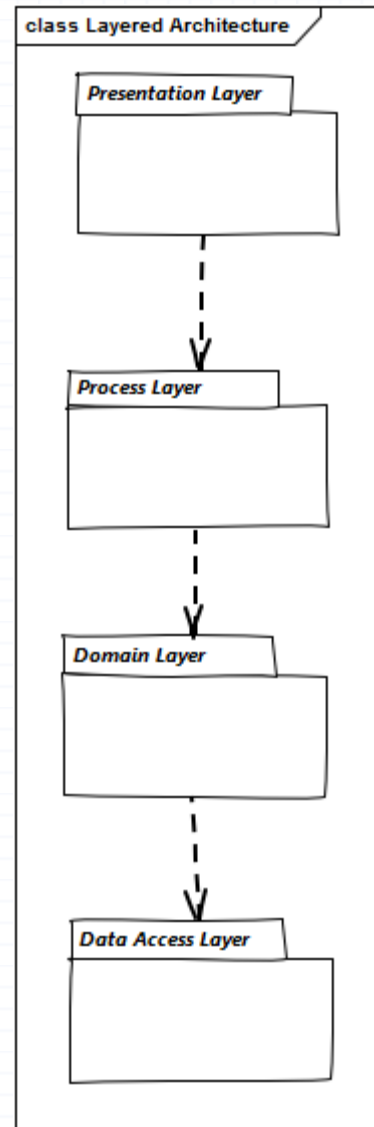
Interface Segregation Principle

"Clients sollten nicht dazu gezwungen werden, von Interfaces abzuhängen, die sie nicht verwenden"

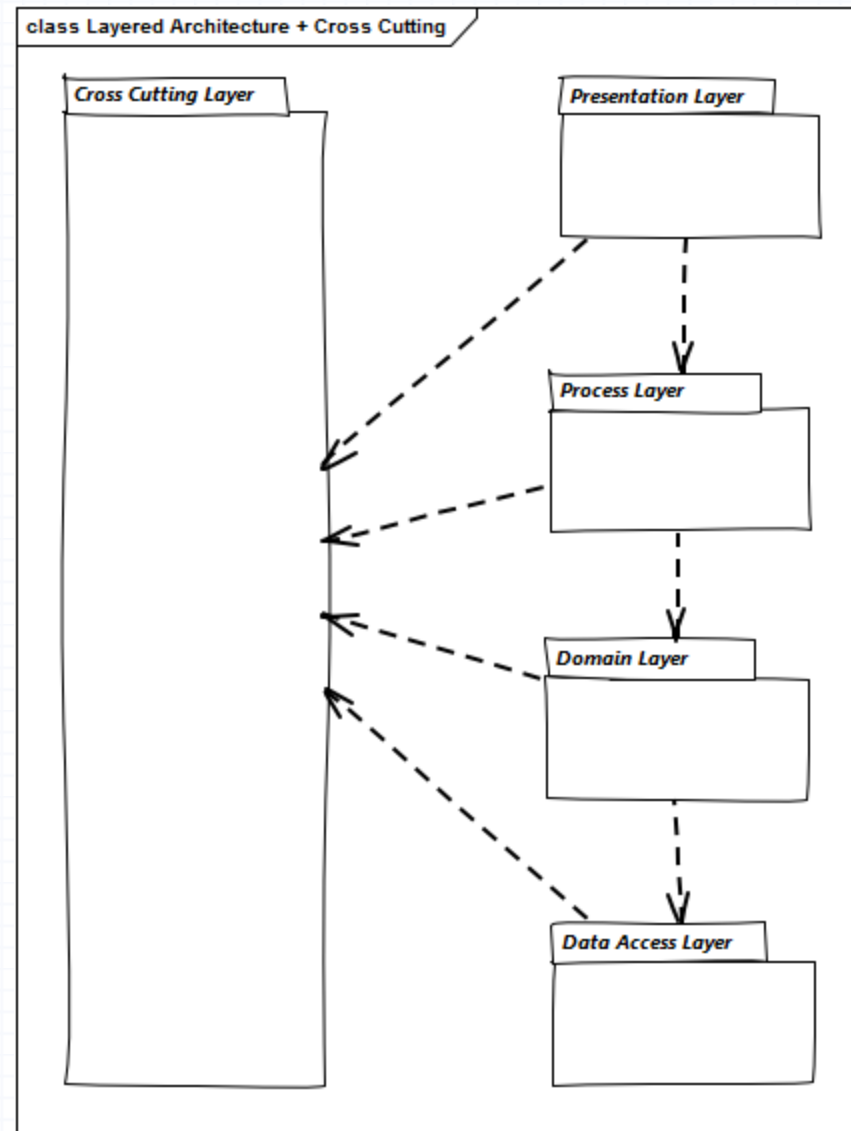
Dependency Inversion Principle

"Abstraktionen sollten nicht von Details abhängen. Details sollten von Abstraktionen abhängen"

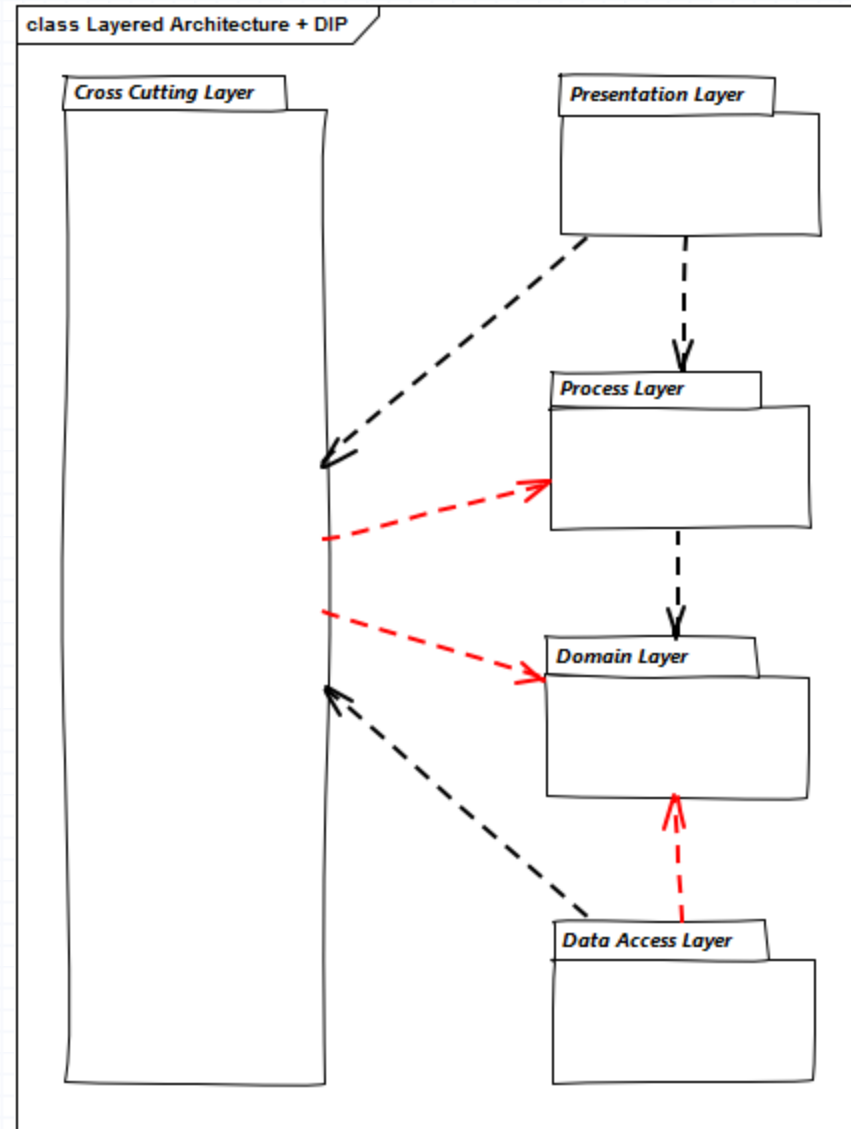
Layered Architecture



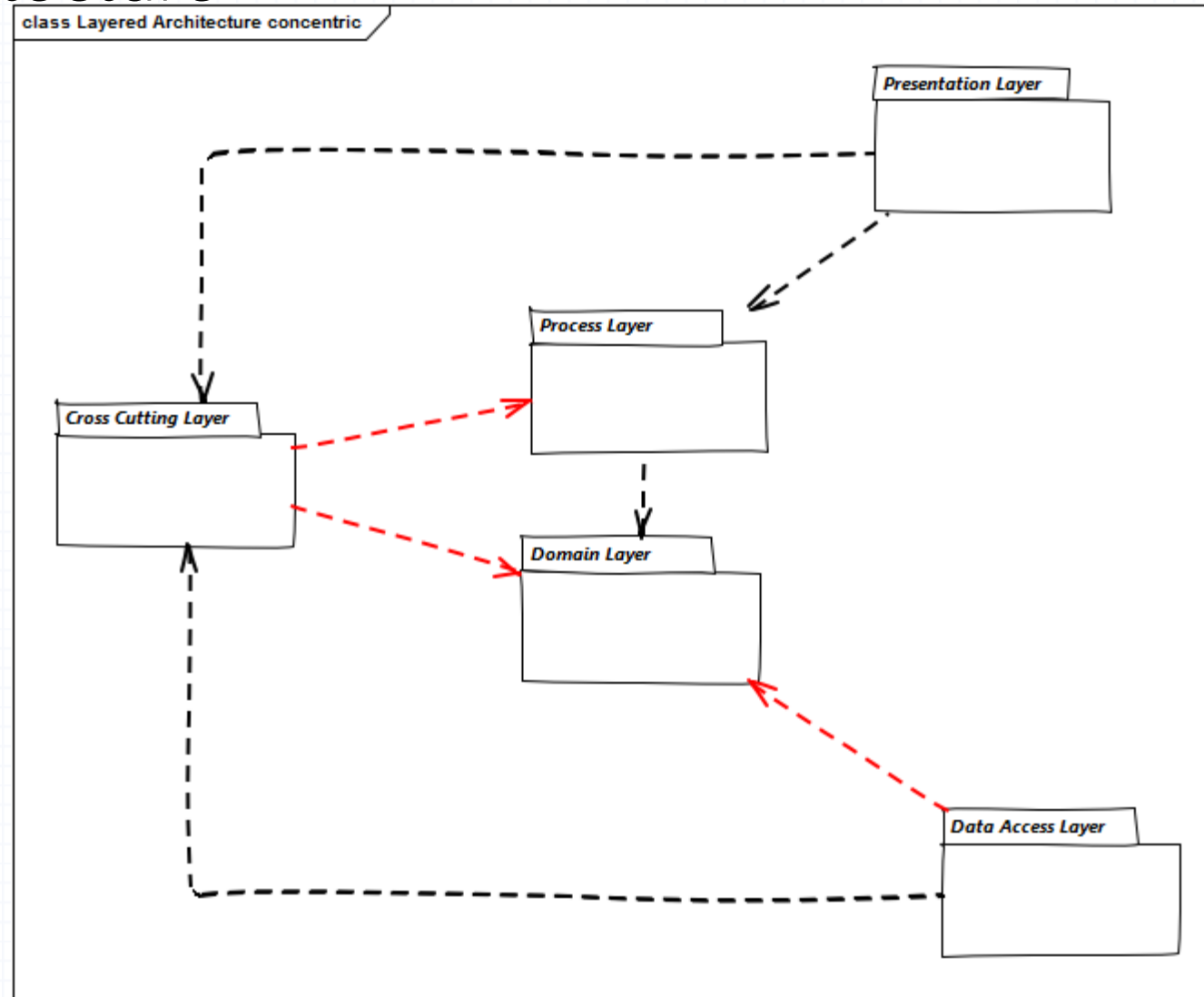
Layered Architecture



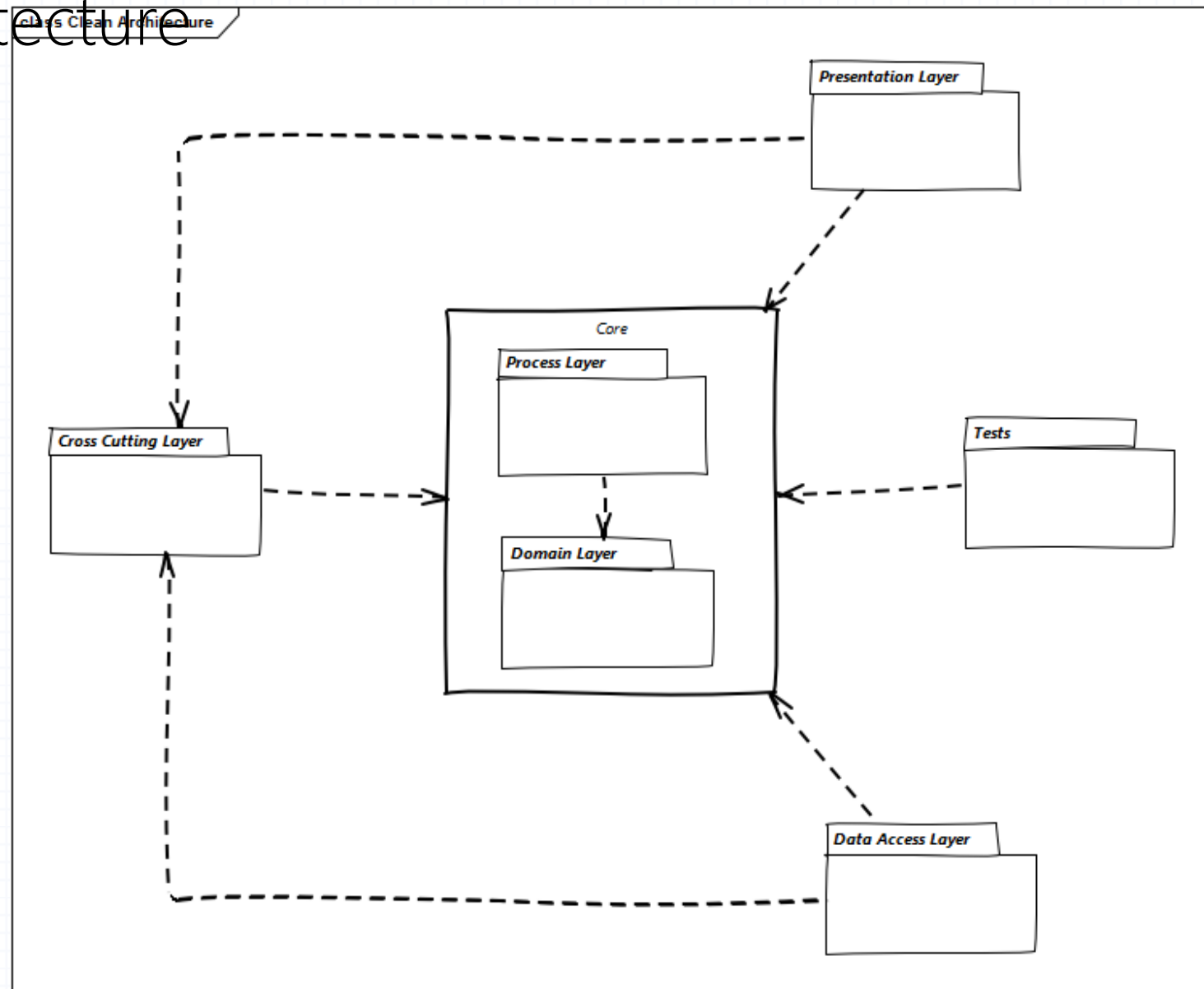
Layered Architecture



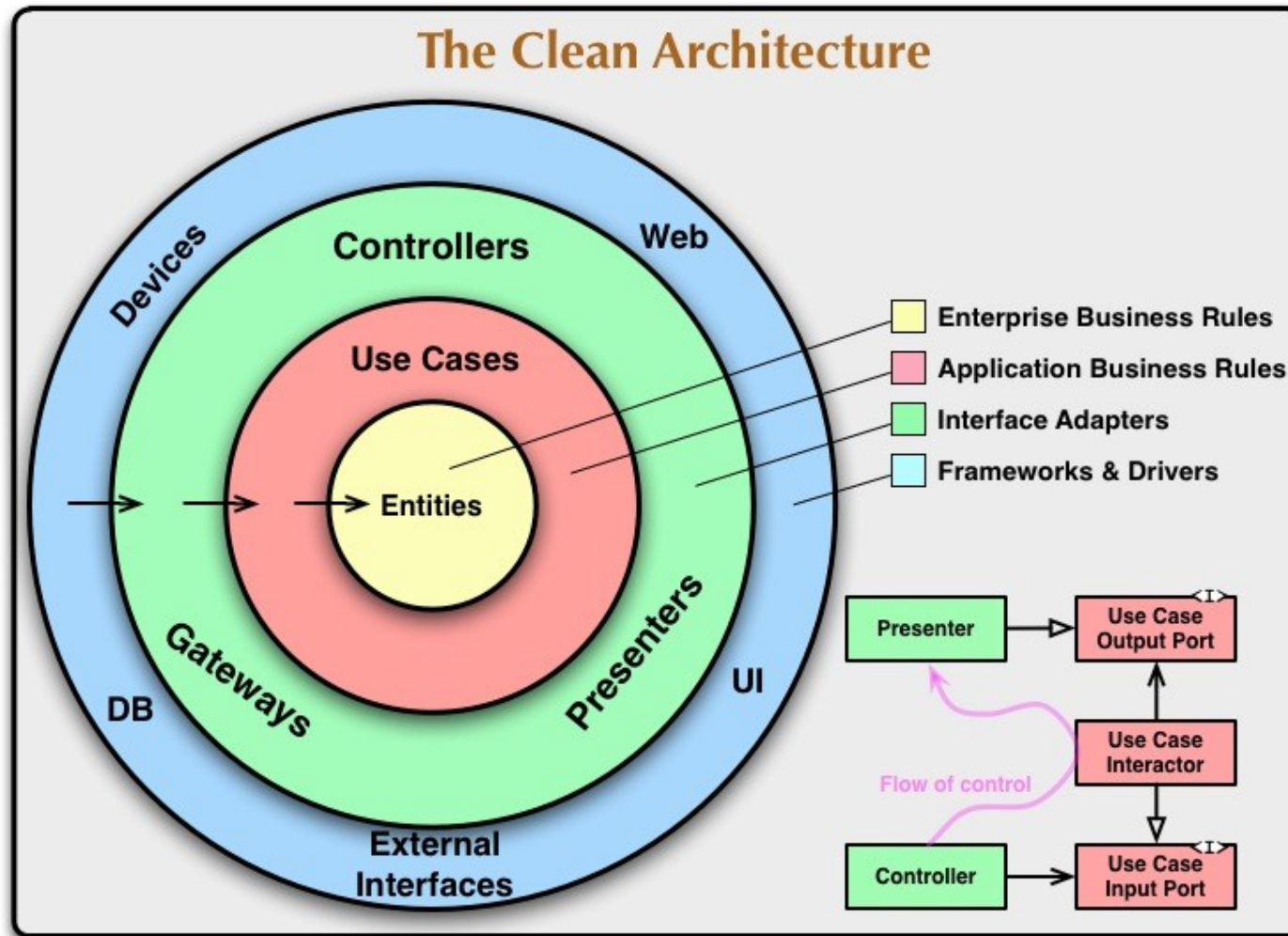
Layered Architecture



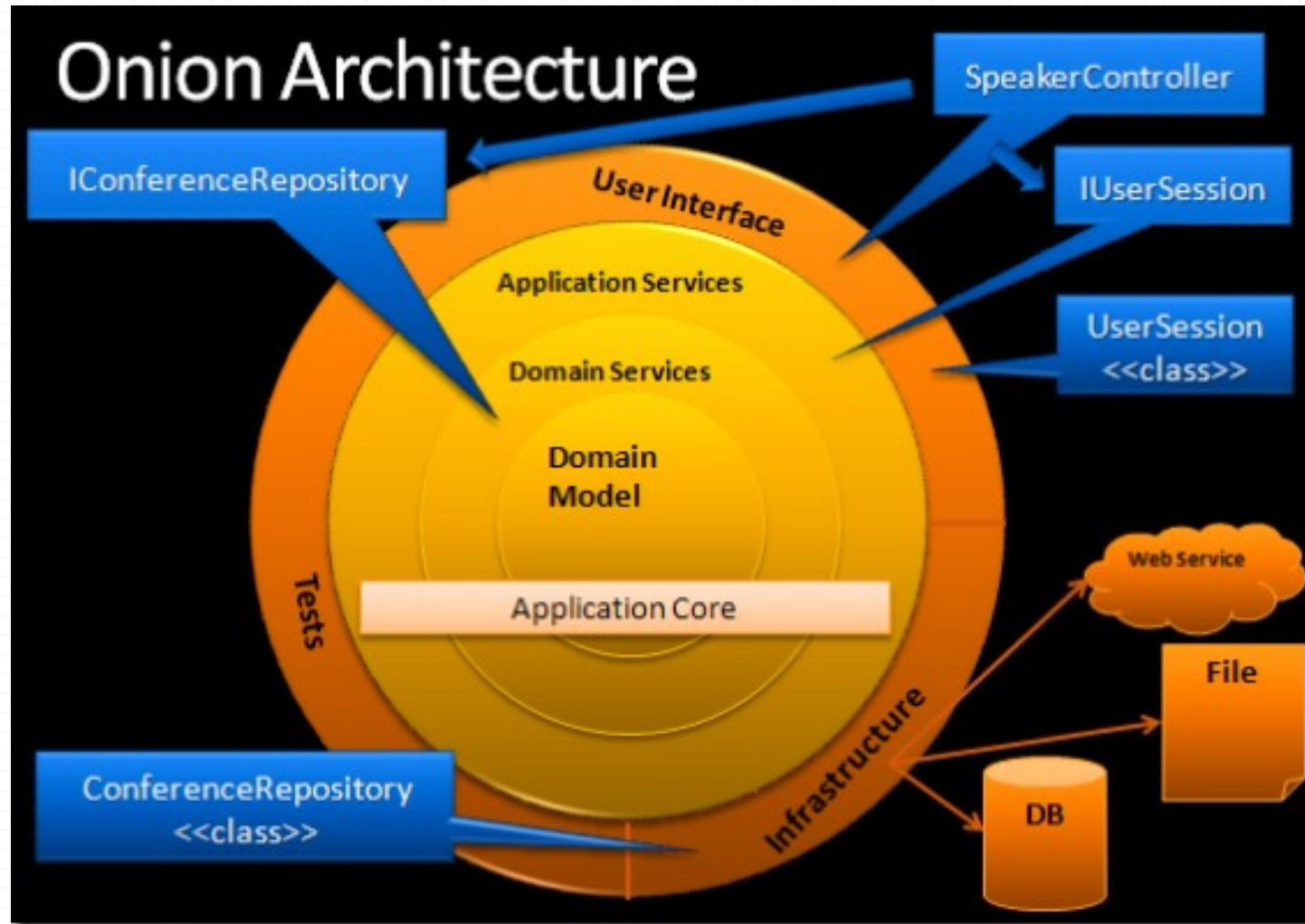
Clean Architecture



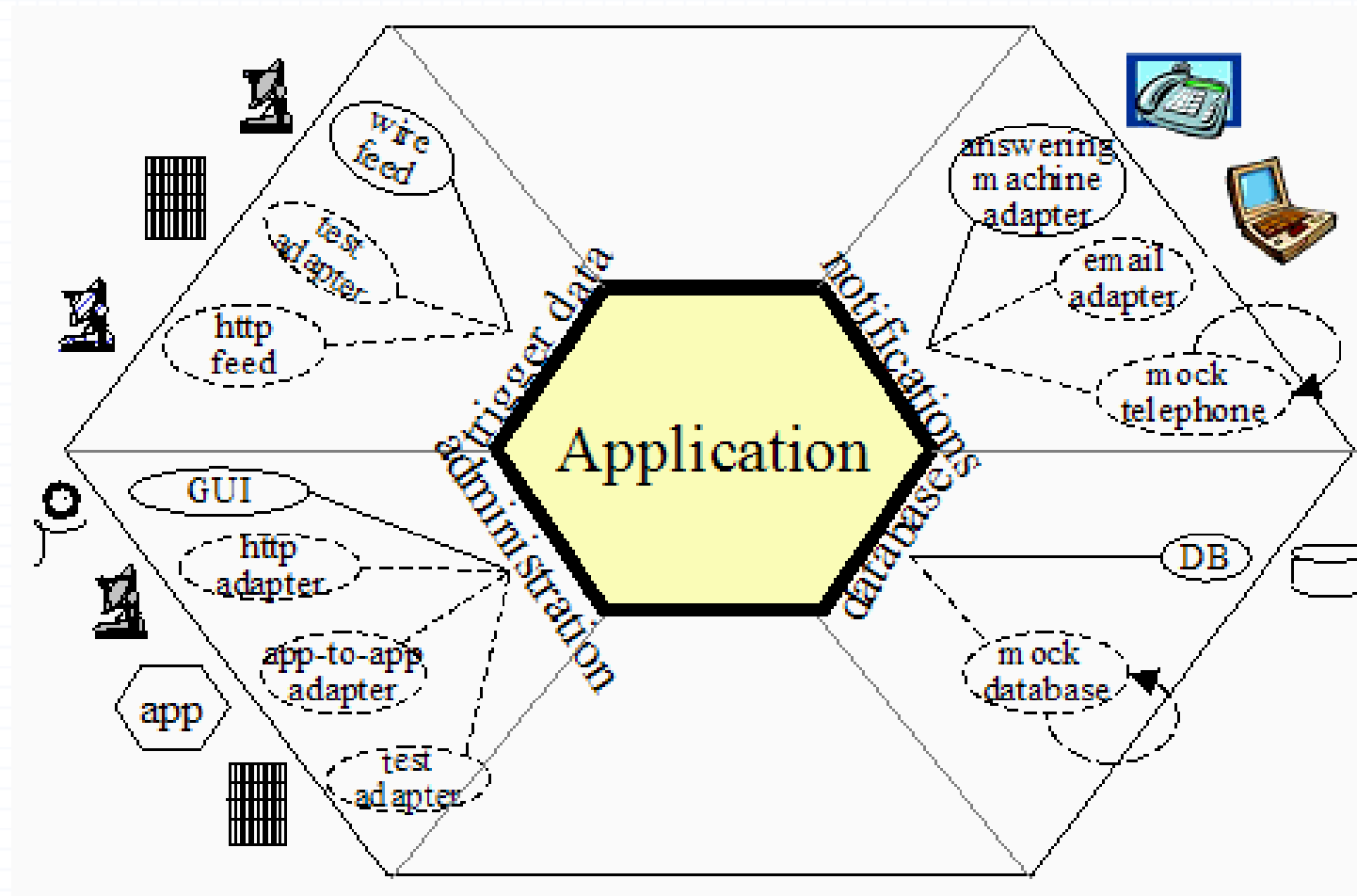
The Clean Architecture



Onion Architecture



Hexagonal Architecture



Demo

Wrap Up - Clean Architecture

Vorteile:

- Layered Architecture + SOLID Principles

- Leicht wartbar, portierbar/austauschbar und testbar


- Wächst mit den Anforderungen

- Gut geeignet für Domain Driven Design

Nachteile:

- Nur bedingt für Datenbank-lastige Anwendungen geeignet

- (z.B. Business Logik aus Performance-Gründen in SQL Stored Procedures)



Q & A + Feedback willkommen

Danke für eure Aufmerksamkeit!

Fragen?

Feedback ist jederzeit willkommen

Twitter: <https://twitter.com/anrichter>

E-Mail: ar@anrichter.net

Xing: <http://xing.to/anrichter>

Links

Die Zwiebelarchitektur und ihre Vorzüge (Daniel Marbach)

<https://jaxenter.de/die-zwiebelarchitektur-und-ihre-vorzuege-19474>

Hexagonal Architecture (Fideloper)

<http://fideloper.com/hexagonal-architecture>

Layers, Onions, Ports, Adapters: It's all the same (Mark Seemann)

<http://blog.ploeh.dk/2013/12/03/layers-onions-ports-adapters-its-all-the-same/>

Exploring the Hexagonal Architecture (Jan Stenberg)

<http://www.infoq.com/news/2014/10/exploring-hexagonal-architecture>

Quellen

Defining Architecture ISO 42010 (ISO-Architecture)

<http://www.iso-architecture.org/ieee-1471/defining-architecture.html>

Softwarearchitektur (Wikipedia)

<http://de.wikipedia.org/wiki/Softwarearchitektur>

Architekturbewertung nach ATAM (Wikipedia)

http://de.wikipedia.org/wiki/Szenariobasierte_Architekturbewertung

SOLID-Prinzipien (Wikipedia)

http://de.wikipedia.org/wiki/Prinzipien_objektorientierten_Designs#SOLID-Prinzipien

The Clean Architecture (Robert C. Martin)

<http://blog.8thlight.com/uncle-bob/2012/08/13/the-clean-architecture.html>

The Onion Architecture (Jeffrey Palermo)

<http://jeffreypalermo.com/blog/the-onion-architecture-part-1/>

The Hexagonal Architecture (Alistair Cockburn)

<http://alistair.cockburn.us/Hexagonal+architecture>