UNIVERSITÄT DES SAARLANDES

Architectural Styles

Architectural Thinking for Intelligent Systems

Winter 2020/2021

Prof. Dr. habil. Jana Koehler

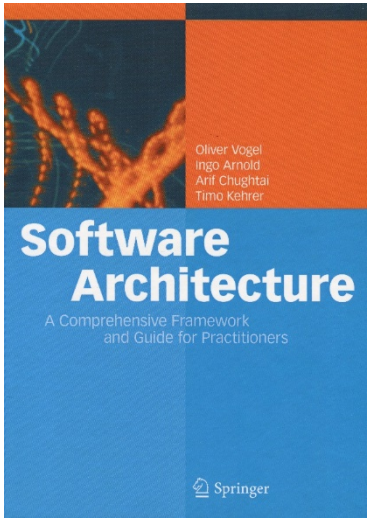
Agenda

- Architectural styles and patterns

- Commonly used architectural styles
  - Layers, Tiers
  - Pipes & Filters
  - Client/Server
  - Peer-to-Peer (P2P)
  - Blackboard
  - Service-oriented architectures (SOA), Microservices
  - Clean/Onion architecture
  - Lambda architecture

Recommended Reading

- Chapter 6.4 (pages 190-231)
  - Subchapters 6.4.12. and 6.4.13 discuss security and cloud architectures and are optional (pages 212-229)



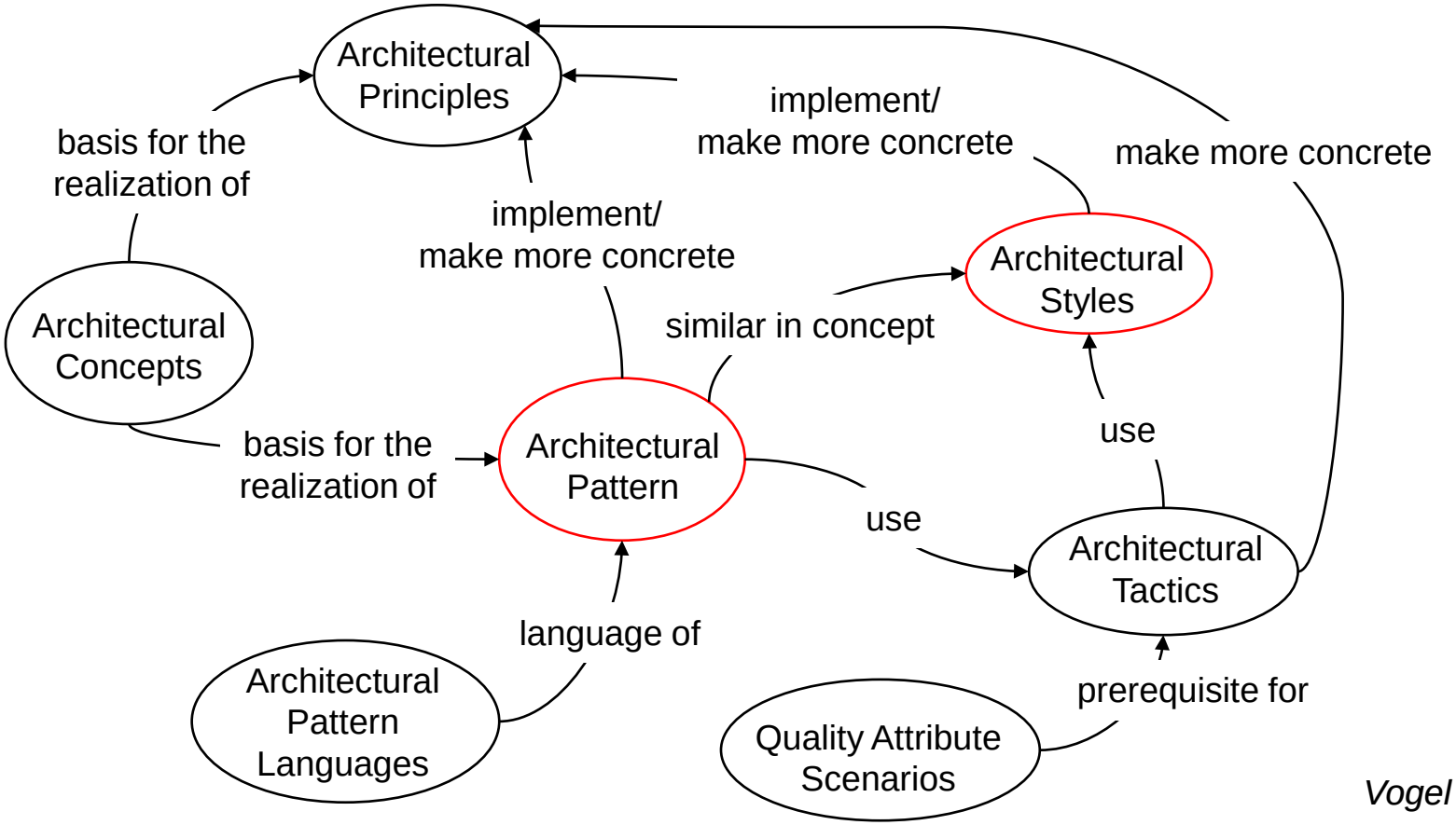
Tutorial Assignment 9:

- We select an architectural style as foundation for the desired system architecture and
- discuss potential alternatives and refinements.

- Views are created to document the architecture at several levels of abstraction.

Importance of Styles and Patterns

Styles and patterns help us to apply proven solutions to our architecture and to implement architectural principles



Vogel et al.

Architectural Pattern

A pattern for software architecture describes a particular recurring design problem that arises in specific design contexts, and presents a well-proven generic scheme for its solution.

Buschmann et al: Pattern-Oriented Software Architecture
Wiley 1996

Architectural Styles

- Architectural style as a pattern for the structural organization of a *family of systems* (Vogel according to Shaw & Garlan)

- Fundamental structure of a software system
  - A set of component types that perform certain functions at runtime
  - A topological arrangement of these components
  - A set of connectors that define the communication and coordination between the components
  - A set of semantic constraints that determine how components and connectors can be interconnected

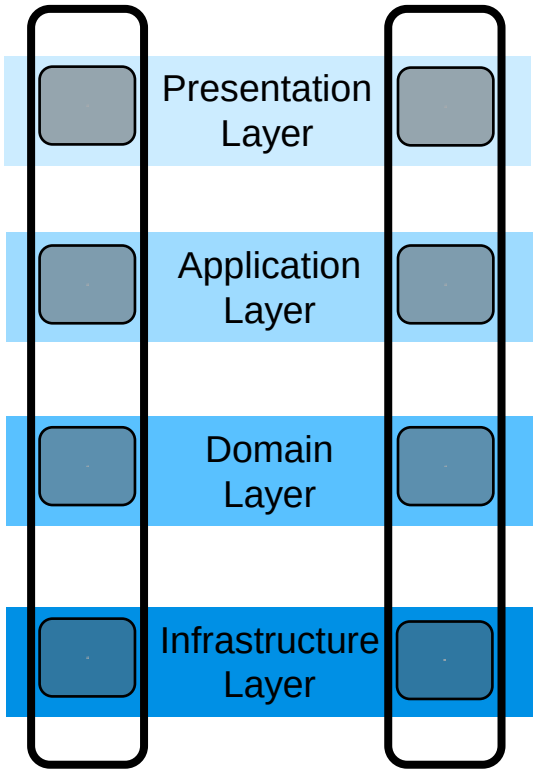
Architecture Styles (Basic Architectural Patterns)

There are different views in the literature, which basic patterns are considered as style. These views also evolve over time …

- We are looking at:
  1. Layers, Tiers
  2. Pipes & Filters
  3. Distributed Systems
     a) Client/Server
     b) Peer-to-Peer (P2P)
  4. Blackboard (originating from AI)
  5. Service-oriented architectures (SOA)
     - Microservices
  6. Clean/Onion
  7. Lambda

1. Layers

- Elements of a layer have a similar degree of abstraction
- A layer offers services to the layer above
  - acts as server
- A layer uses only the services of the layer directly below
  - acts as client



➢ Higher layers accessing deeper layers (below the immediate neighbor) destroy the architecture!

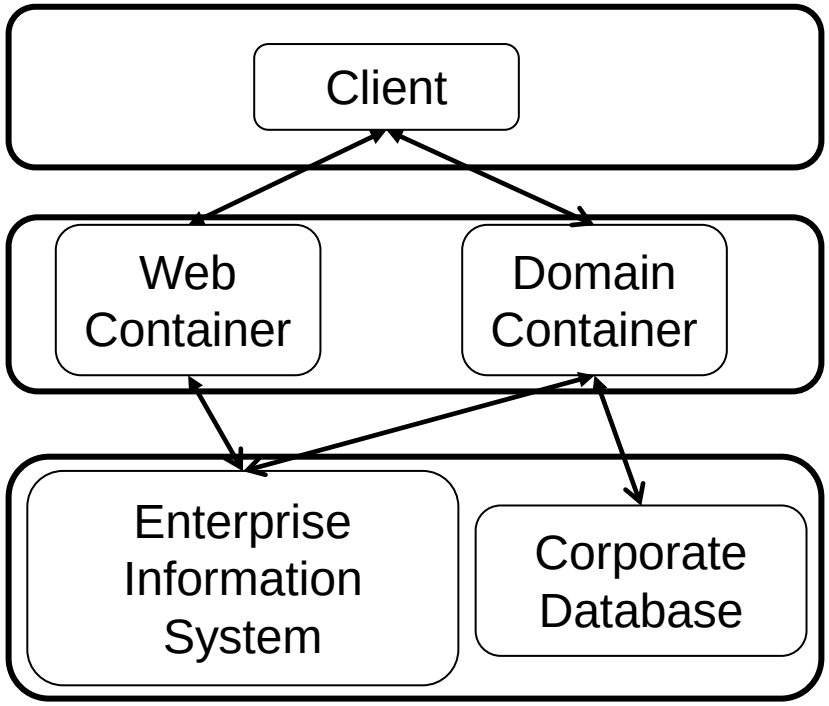
Advantages and Disadvantages of Layers

+ Easy to understand structural concept
+ Minimizes dependencies between components
+ Layers are independent of each other during development and operation
+ Changes in one layer affect at most the lower neighboring layer

– Performance of the system can be negatively affected if requests have to be forwarded through multiple layers
– Changes in the data model can affect all layers (data management in the infrastructure, domain, application, presentation layers)

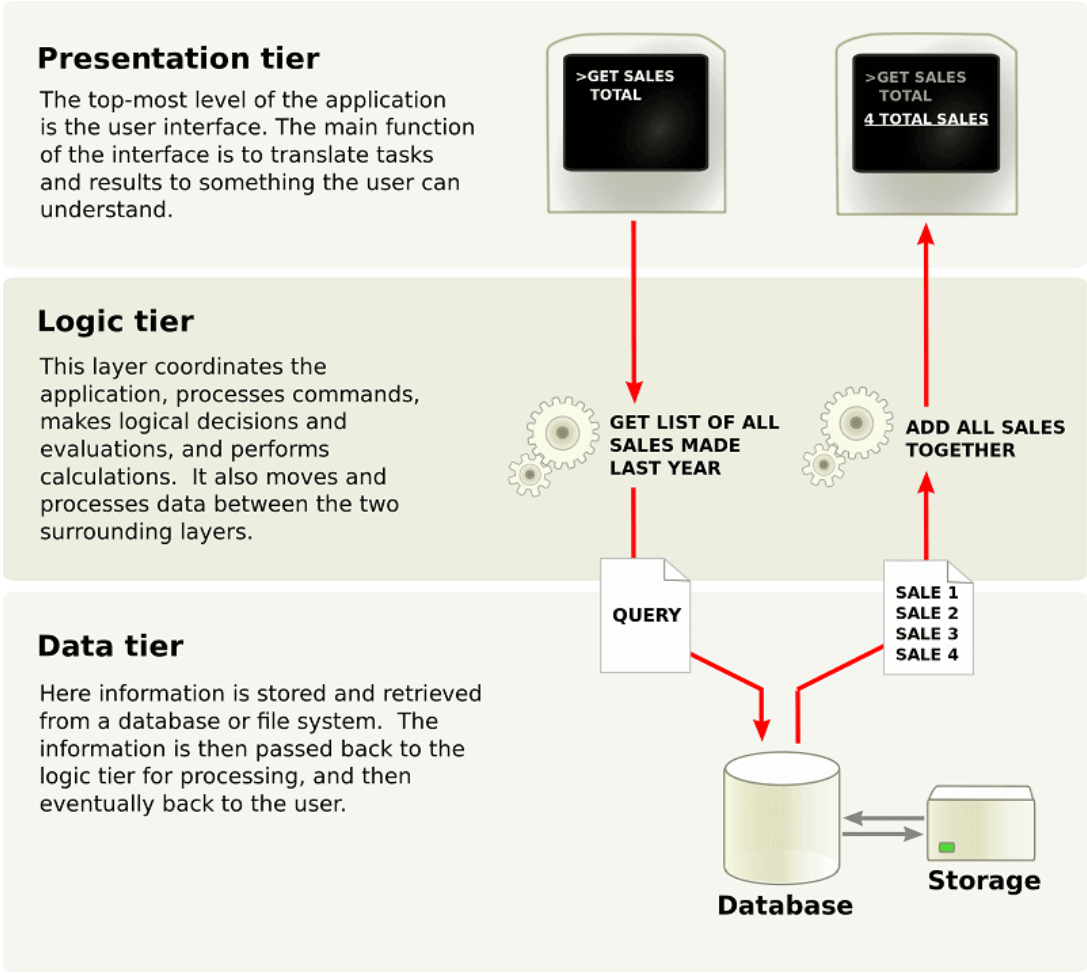
Special case of Layers: N-Tier Architectures

- Can be seen as a special form of layering, but also as a specific form of a client/server architecture

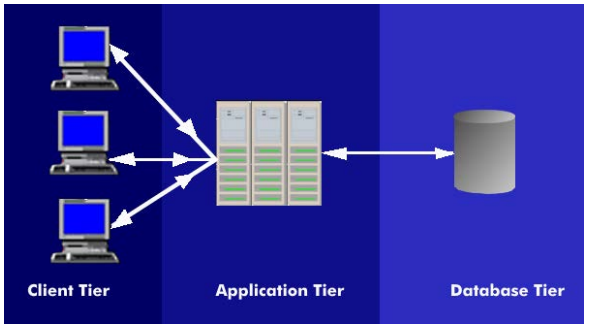


- Tiers can communicate more flexibly with their neighbors
- Bidirectional dependencies violate top-down principle of layer architecture
- *Strong* (access to immediate neighbor only) or *flexible* separation of layers (access across arbitrary tiers)
  - No dependency of lower tiers on presentation tier
- Integration effort potentially higher

Example: 3-Tier Architecture



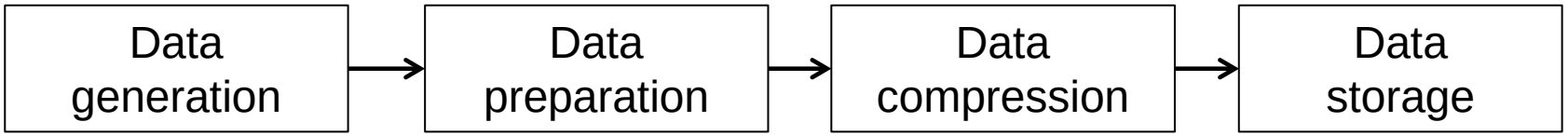
https://stackify.com/n-tier-architecture/



https://www.itwissen.info/Three-Tier-Architektur-three-tier-architecture.html

2. Pipes and Filters

- Sequence of processing units (filters) connected to each other by data channels (pipes)
  - Each filter passes its result directly to the next filter
  - Pipes act as connectors between filter components



- Various coordination models can be implemented
  - Decentralised/centralised control
  - Pipes passive or active
  - Data transfer complete, piecemeal, time-shifted

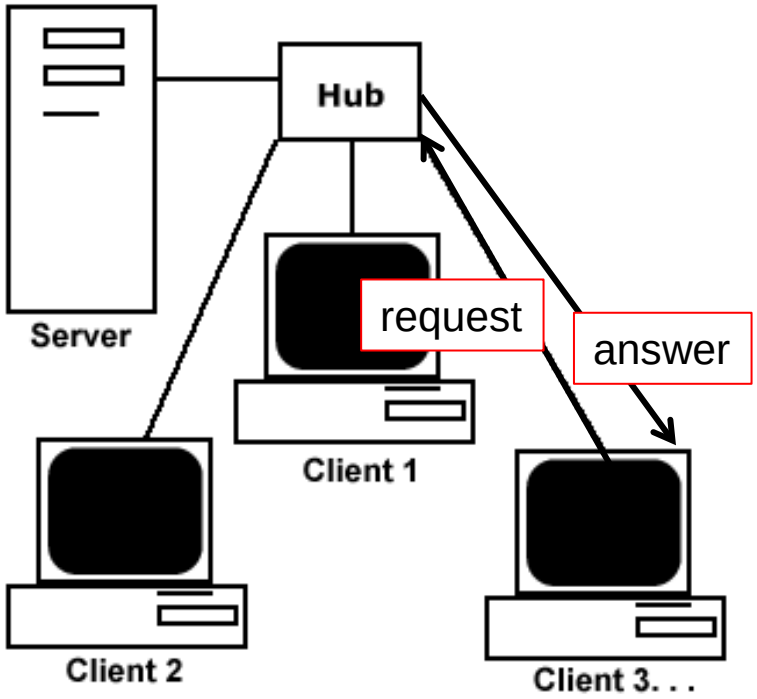
Advantages and Disadvantages of Pipes and Filters

+ Easy to implement
+ Easy-to-understand structure
+ Clearly structured flow of information and control
+ Powerful pipes can decide to which instance of a filter (load balancing) or to which filter component (encapsulation) they pass the data

– Filters do not know each other
  – Subsequent errors cannot be recognized and handled
– Configuration of sophisticated processing chains can be difficult
– Filters only communicate via data objects
  – All processing information must be contained in the data or in the central processing unit

3a. Client/Server

- Applications (clients) are operated locally
- Services requested by clients are centrally managed and made available via a server
- Communication is a simple request-response scheme



Rich versus Thin Client

- How is the functionality distributed between client and server?
  - Thin Client
    - Only limited functionality directly implemented in the client, highly dependent on server functionality
      - Gmail (web browser + web server)
  - Rich Client
    - A lot of functionality locally in the client, less dependency on the server
      - MS Outlook (Windows Application + Mail Server)

Advantages and Disadvantages of Client/Server

+ Centralization of important, compute-intensive or sensitive computations on the server
+ Thin clients easy to deploy and maintain
+ Rich clients still usable in case of server failure

– High network load (especially with thin clients)
– Distribution of functionality not always easy to decide
– Limits of scaling with very high numbers of clients (and a single server)
  – Scaling is hardly an issue in modern virtualized infrastructures

3b. Peer-To-Peer (P2P)

- Equal components (peers) distributed over a network that perform the role of both client and server and share resources
- One type of connector («inter peer connection»)
  - Usually the internet
- No centralized control, peers are free in communication (any-to-any)
- Localization of peers by
  - decentralized communication (peers exchange individual lists of known peers with each other) or
  - a central service (registry)

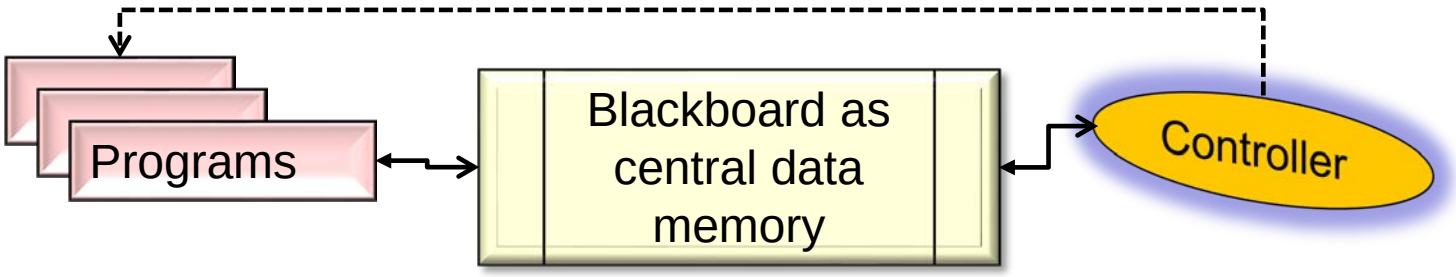
Advantages and Disadvantages of P2P

+ High reliability (no single point of failure)
+ Calculation-intensive tasks can be distributed
  - E.g. Seti@Home

– Finding and detecting peers in large networks can be difficult without a centralized registry
  - Potential danger of P2P network decomposition
– No obvious solution how to implement error handling
  - Who reacts when a peer malfunctions?
– No guaranteed response times

4. Blackboard

- Originally from Artificial Intelligence to solve complex problems for which no deterministic solution method exists
- Blackboard acts as shared memory holding information about the state of the problem
- Collaborative problem solving through otherwise independent programs (agents)
  - No calls between the programs, communication only via blackboard
- Optional control component evaluates solution progress on the blackboard and activates available programs

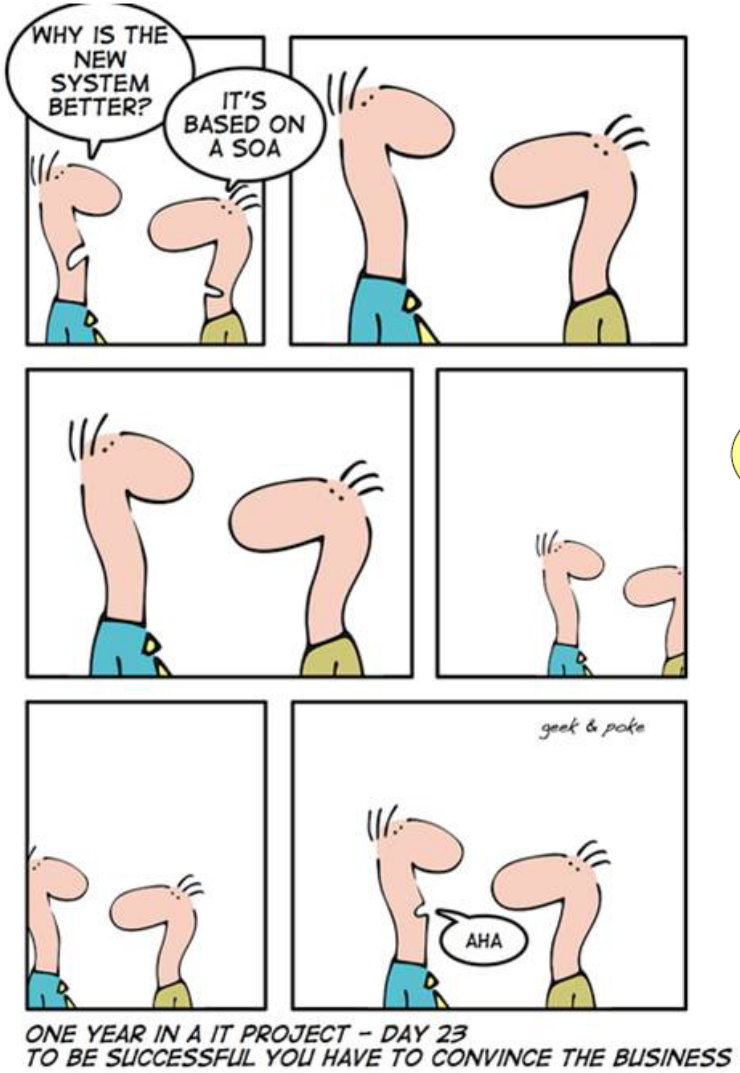


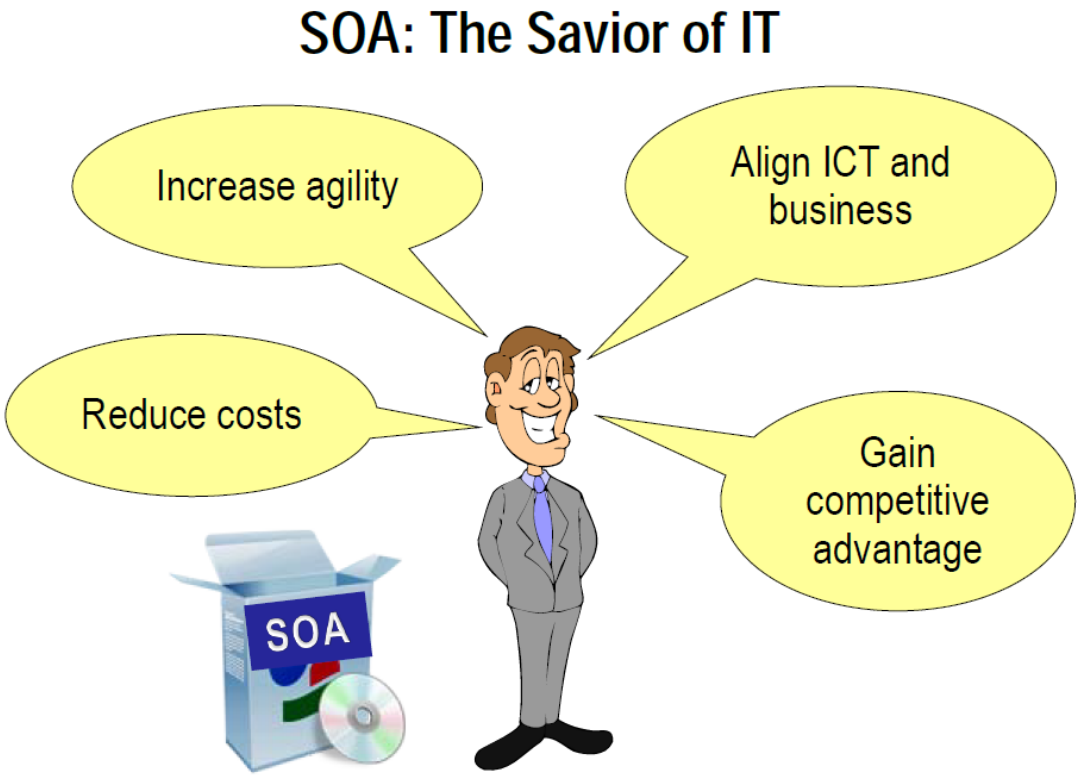
Advantages and Disadvantages of Blackboard

+ Simple integration of complex systems
+ Parallel computations by agents possible
+ Agent components can be easily reused in other systems
+ Scalable
+ Robust

– No guarantee of finding a solution
– Finding the right control strategy is difficult
– No guaranteed response times and solutions
– Difficult to test

5. Service-Oriented Architecture (SOA)



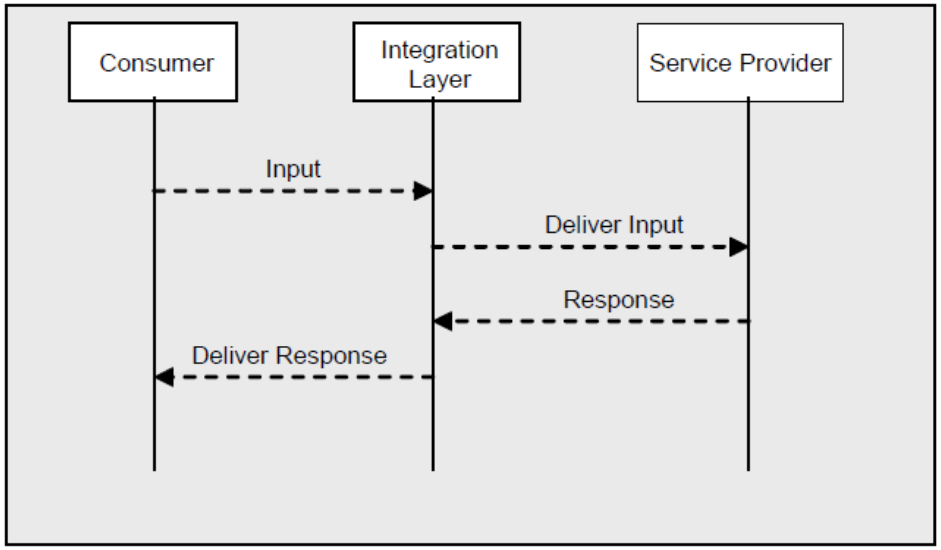


- Task for Business + IT

Quelle: Gartner

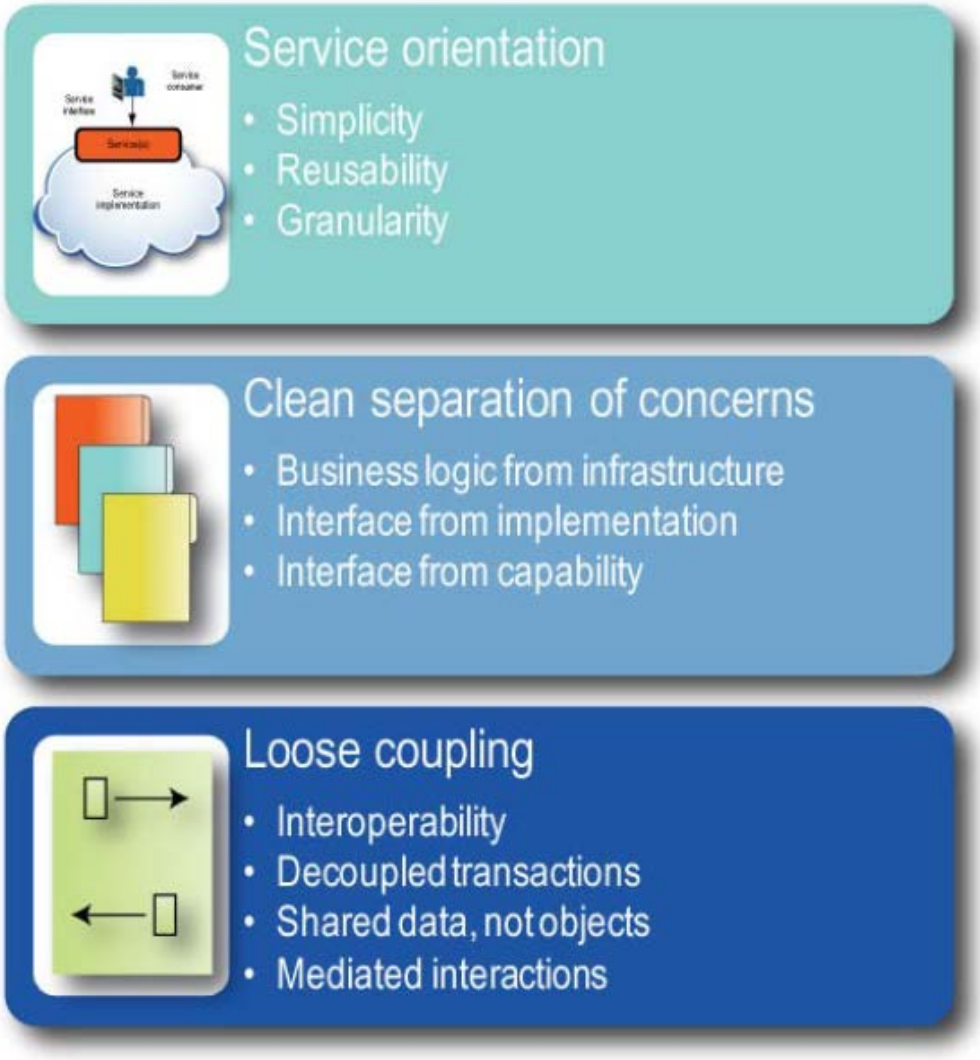
SOA from an Architectural Perspective

- Specification of services, data formats (messages) and communication protocols
- Applications as orchestration of services to achieve specific business goals
  - Service provider and service consumer



- http://soa-manifesto.org

Basic Principles and Technologies

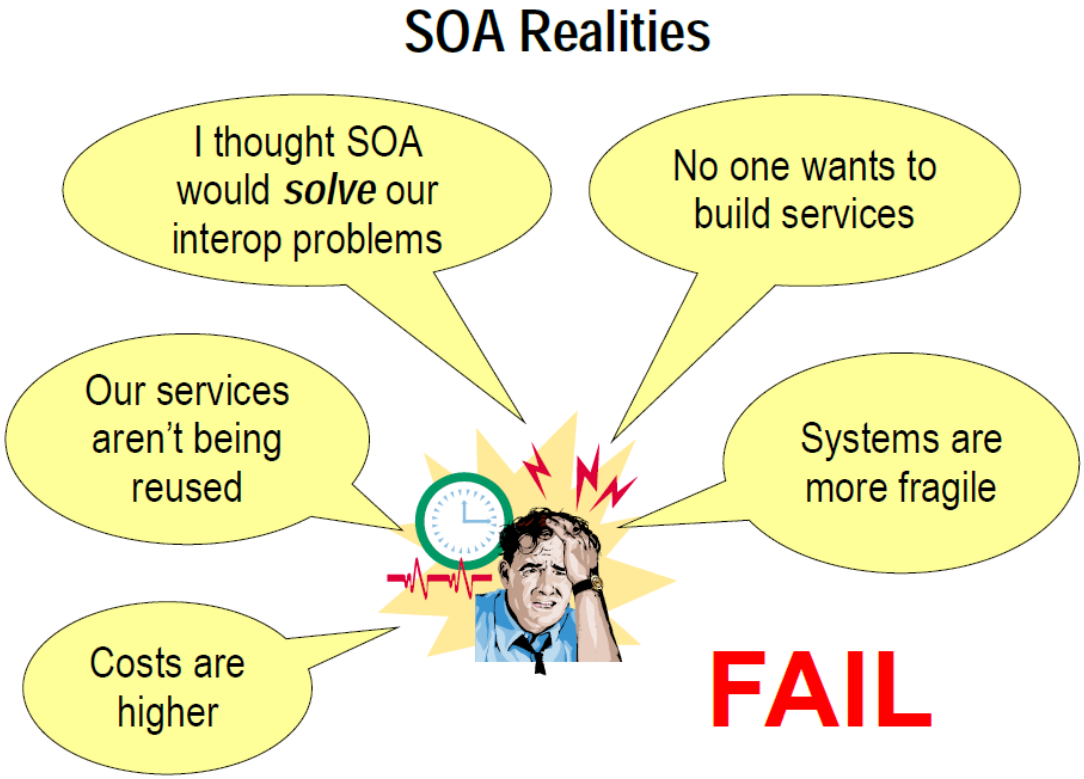


A.Thomas Manes, Gartner/Burton

- Wide range of standards
  - XML-based data/message formats
  - SOAP/REST as the most important protocols
  - Web services (WSDL)
  - Microservices/Container
- Service-oriented system
  - Modular
  - Distributed
  - Traceable
  - Replaceable
  - Re-usable

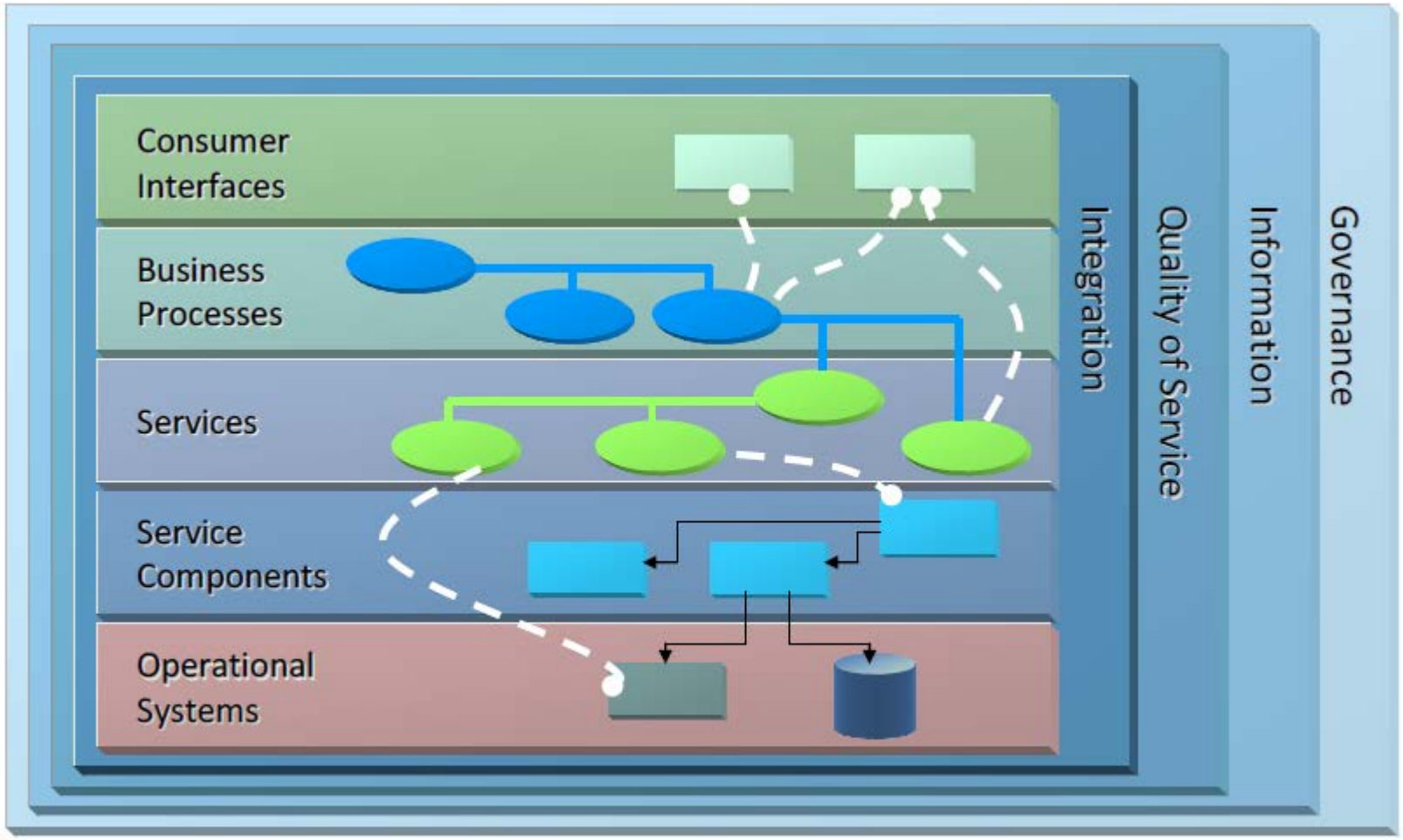
SOA Governance

- SOA governance crucial for successful implementation



A.Thomas Manes, Gartner/Burton

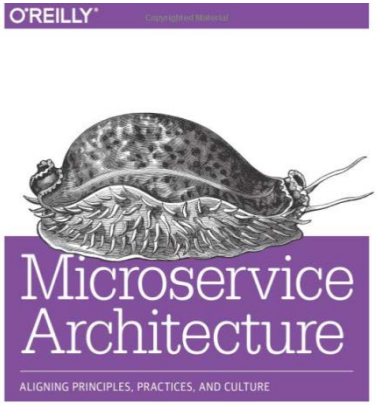
Open Group SOA Reference Architecture



(C) The Open Group 2009

Microservices

- Software applications as suites of independently deployable services
- Based on
  - business capability
  - automated deployment
  - intelligence in endpoints
  - decentralized control of technologies and data
- Communication only through web service APIs
- Services evolve independently from each other without coordination

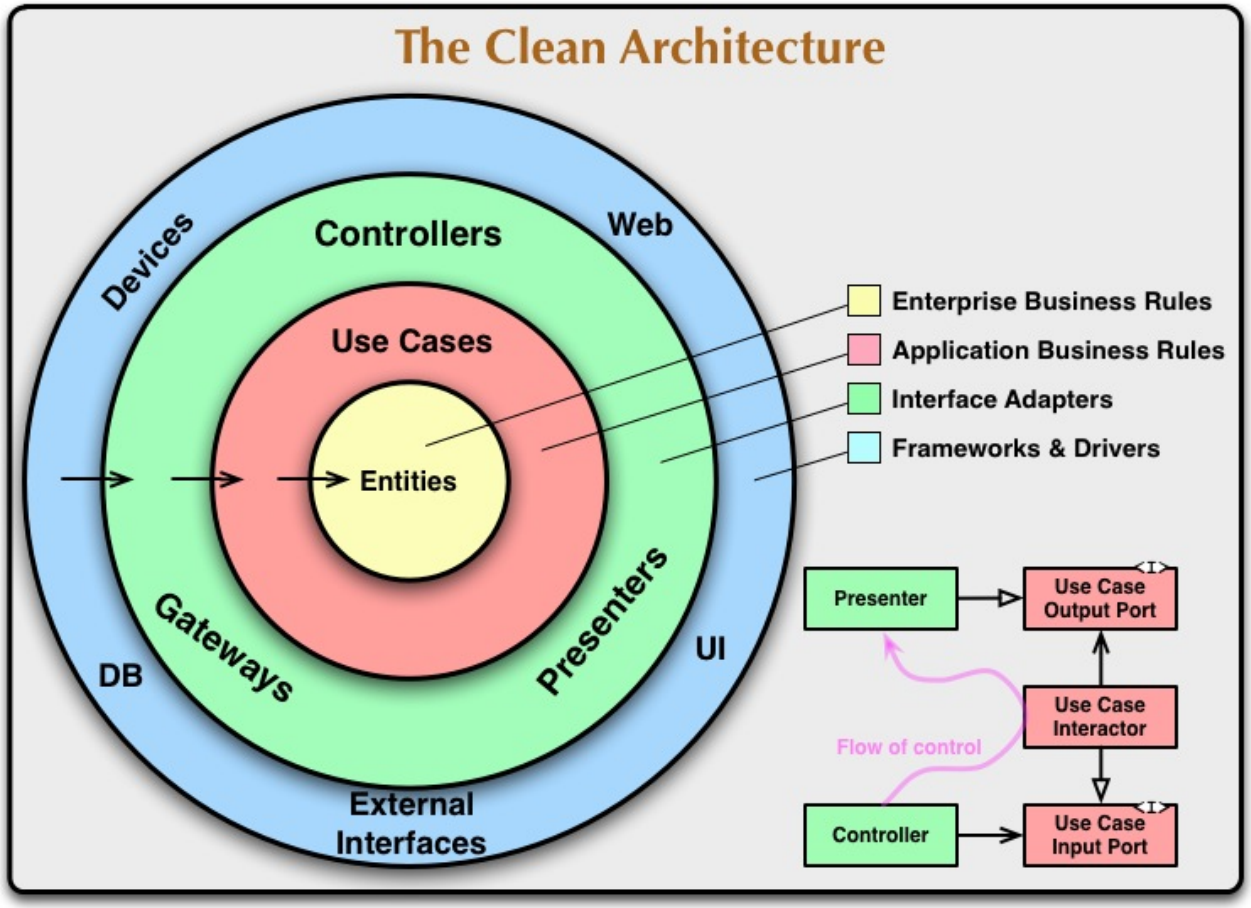


"Truly implement SOA and decouple services"

Advantages and Disadvantages of SOA

- \+ Very flexible architectural style with simple basic model
  - \+ System functionalities encapsulated as reusable assets
  - \+ Binding of services at runtime and lookup in registries possible
- \+ Wide range of fully-developed standards
- \+ Brings together Business and IT
- \+ Prerequisite for Cloud, Mashups, …
- – Inherent complexity of open, decentralized systems
- – Multitude of difficult questions
  - – Service design, interoperability, standards, virtualization
- – Testing can be difficult, implicit dependencies can make architectures fragile

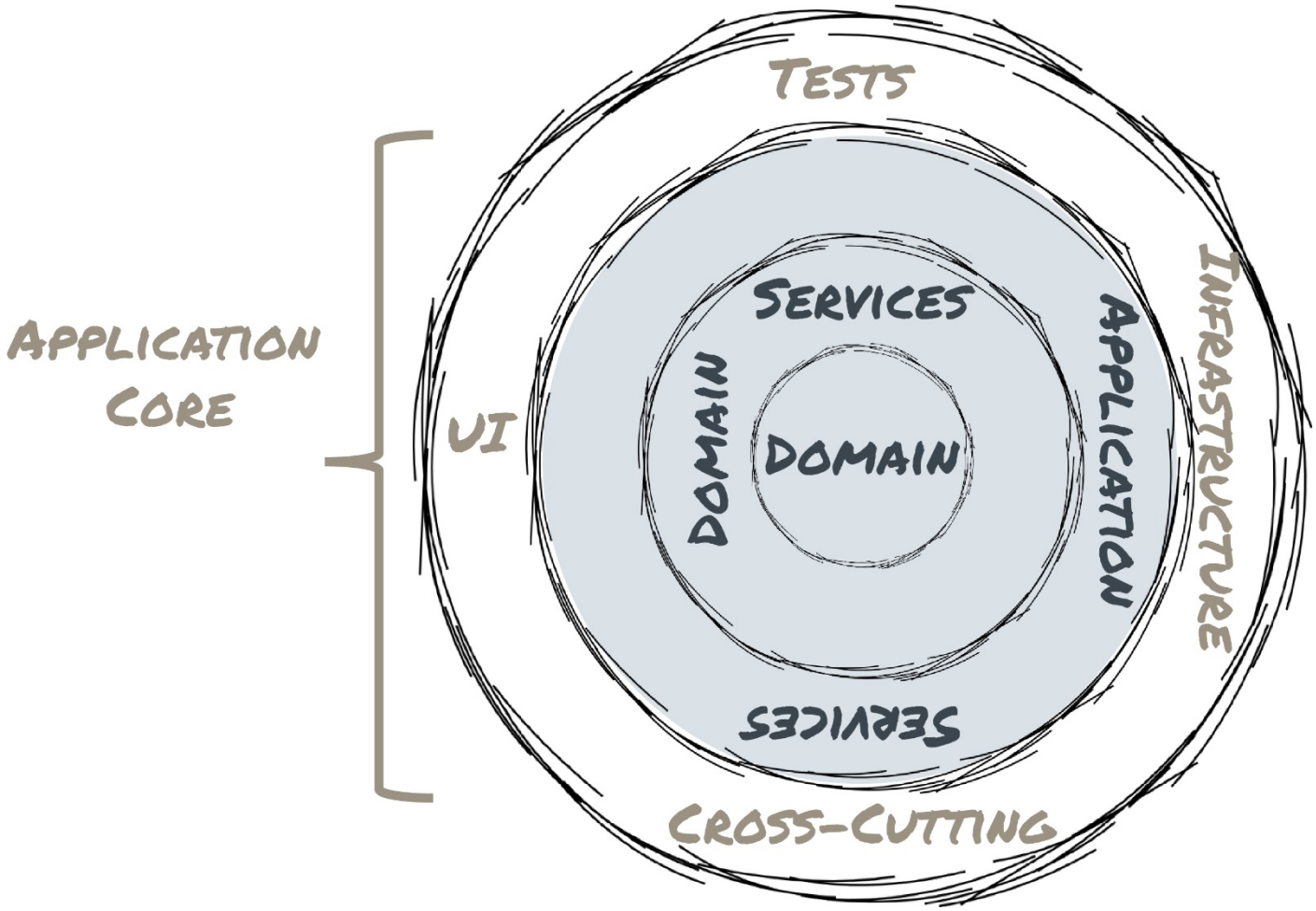
6. Clean = Hexagonal = Onion = Ports and Adapters



Onion Architecture (Jeffrey Palermo)
Ports and Adapters (Alistair Cockburn),
Screaming Architecture (Robert C. Martin),
Data Context Interaction DCI (James Coplien, Trygve Reenskaug)
Boundary Control Entity (Ivar Jacobson)

http://alistair.cockburn.us/Hexagonal+architecture
Bild von http://blog.8thlight.com/uncle-bob/2012/08/13/the-clean-architecture.html

Onion and Domain-Driven Design



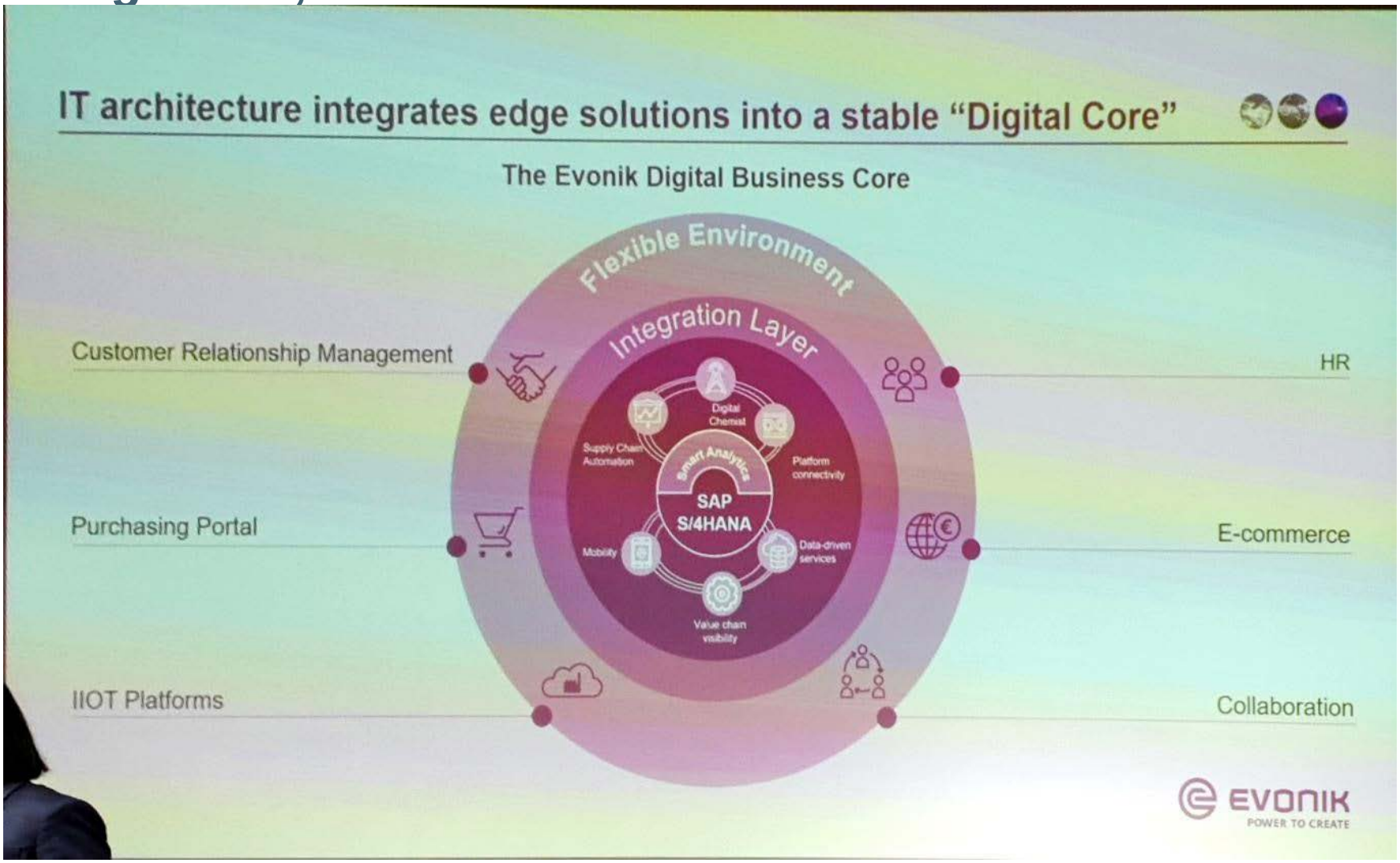
https://jaxenter.de/wp-content/uploads/2015/05/marbach_zwiebelarchitektur_6.jpg

Advantages and Disadvantages of Onion Architecture

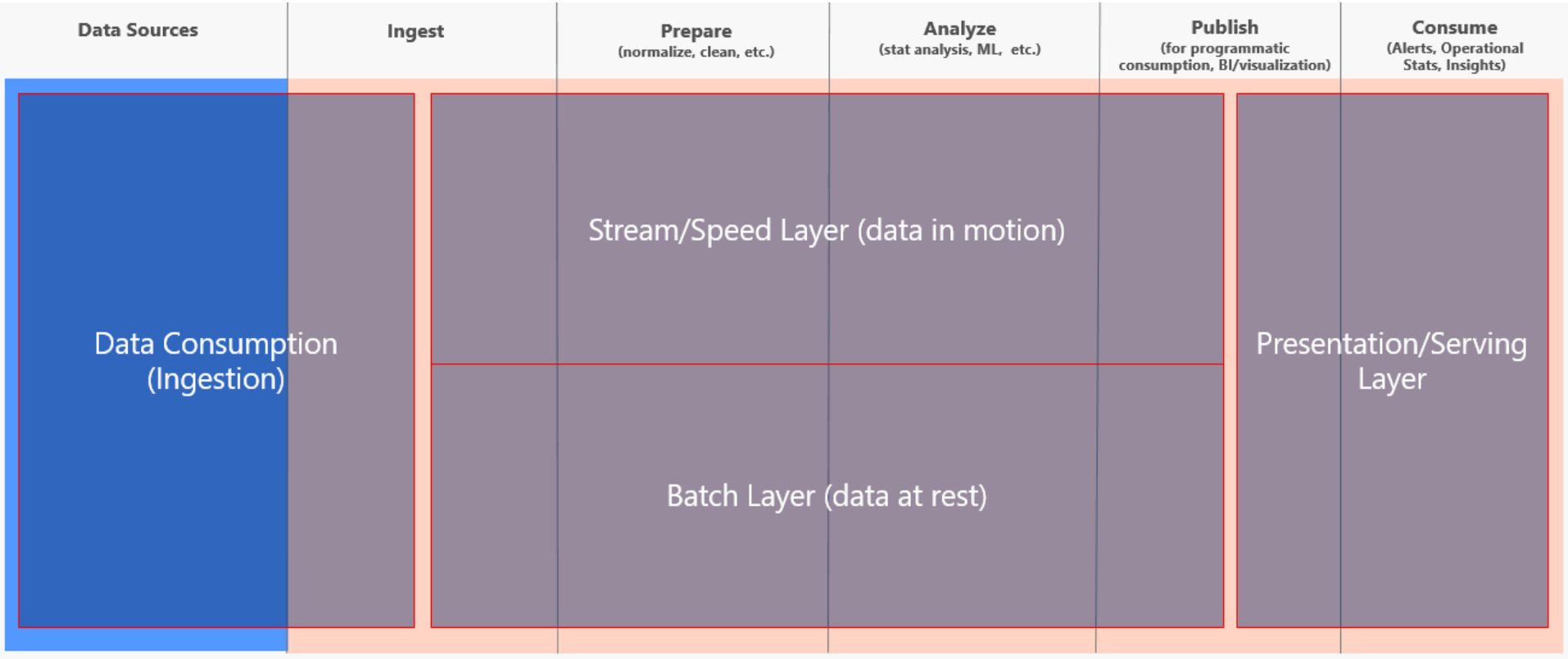
+ Puts data in the focus of attention to reduce coupling
+ Well suited to implement bounded contexts and domain-driven design
+ Focus is on applications and services
+ Enables multi-channel user interfaces

– Difficult to understand and apply
  - Various names suggest many sources for confusion
  - Is it really different from SOA?
– No dedicated technology stack (?)
– Too much variance to be a good pattern - needs further work in my eyes

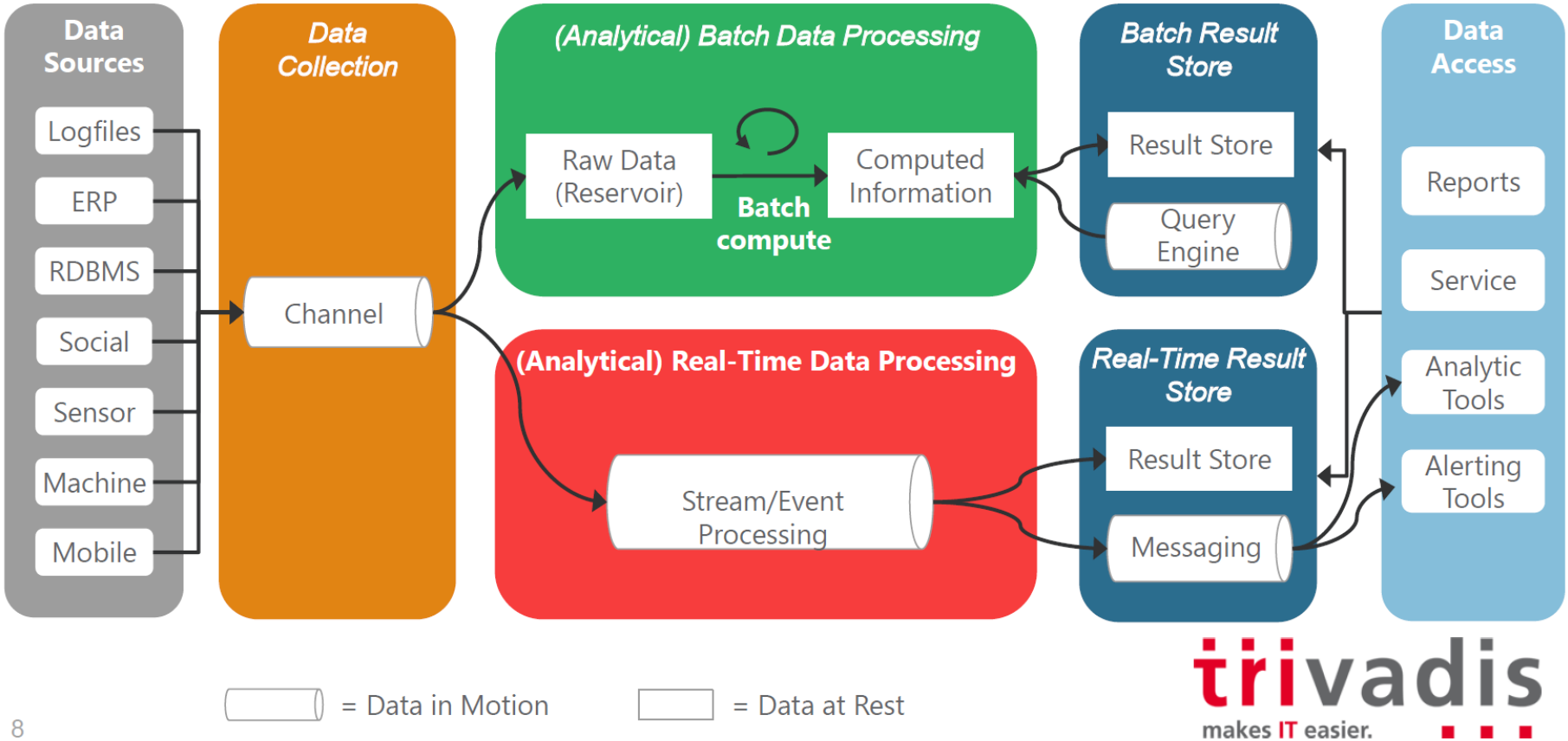
Example Onion presented at Conference on Strategic IT Management, Munich 2020



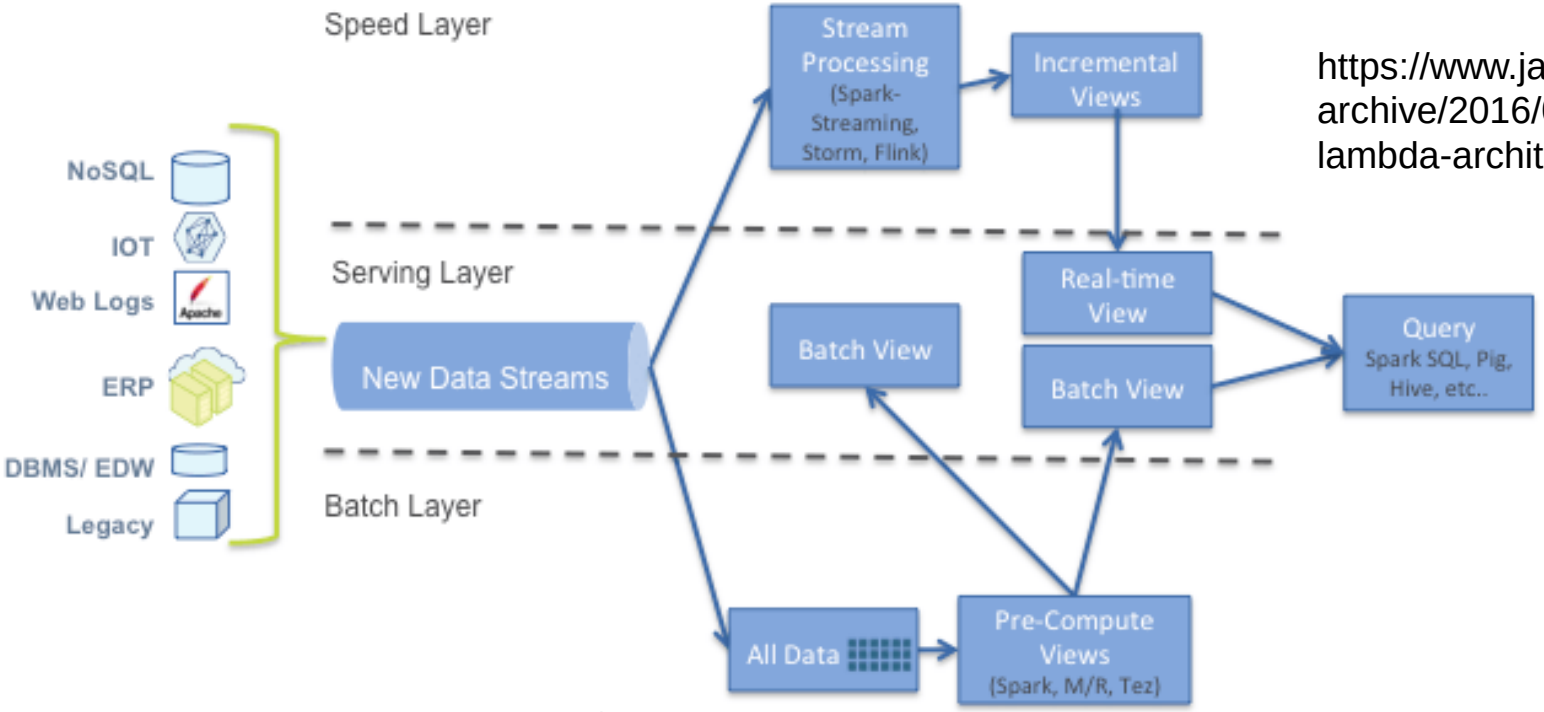
7. Lambda Architecture for Big Data



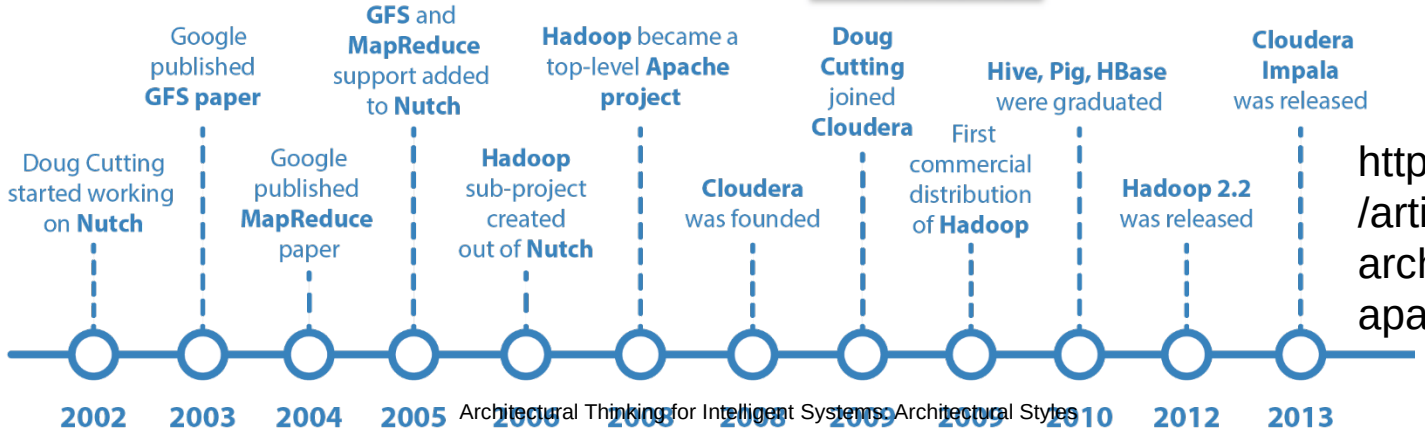
A More Detailed Picture of Lambda



Implementing Lambda Based on Apache Hadoop

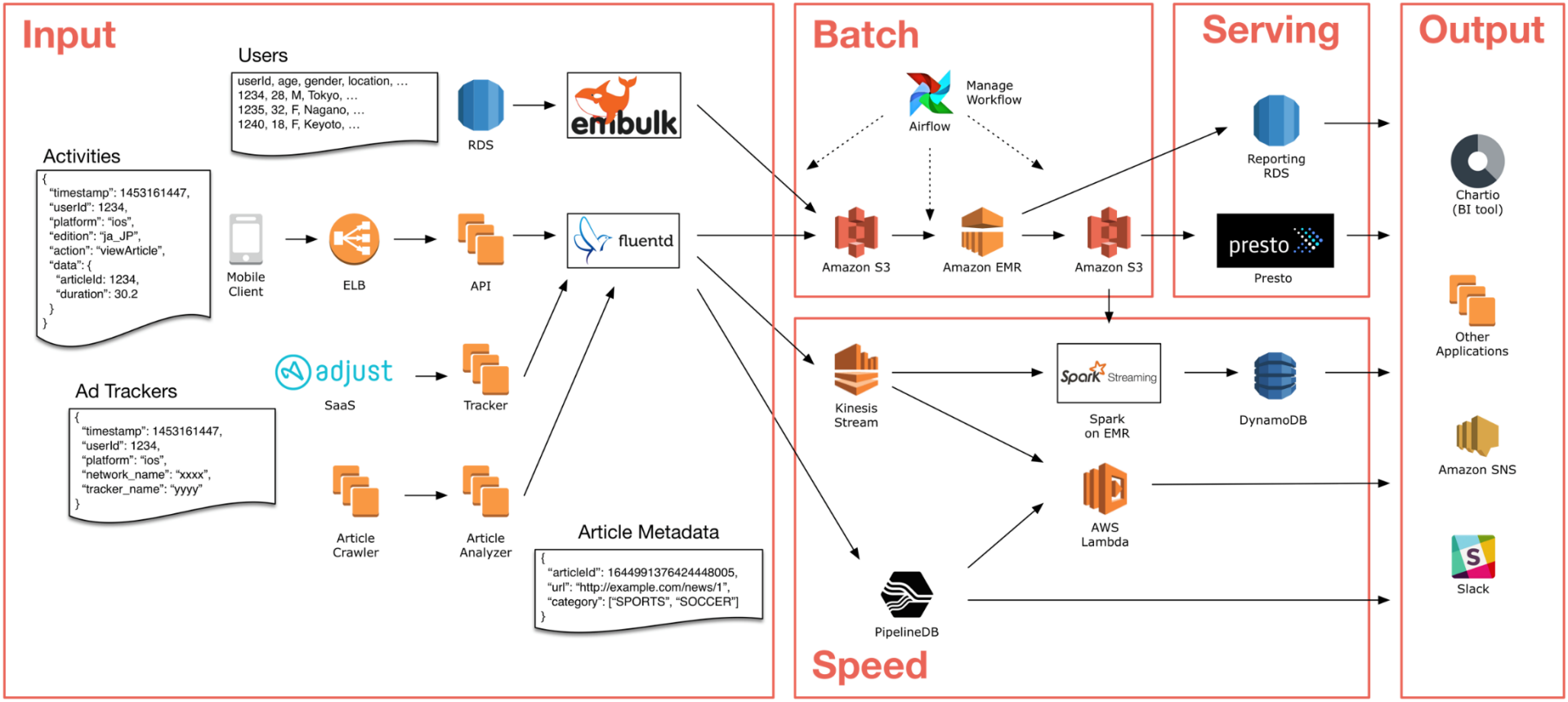


https://www.jamesserra.com/archive/2016/08/what-is-the-lambda-architecture/



https://dzone.com/articles/lambda-architecture-with-apache-spark

Example on AWS



https://aws.amazon.com/de/blogs/big-data/how-smartnews-built-a-lambda-architecture-on-aws-to-analyze-customer-behavior-and-recommend-content/

Advantages and Disadvantages of Lambda

+ The de-facto style for Big Data applications
+ Clean separation of streaming and batch data
+ Well-proven process phases to work with data
+ Well-established and strongly advancing technology stack

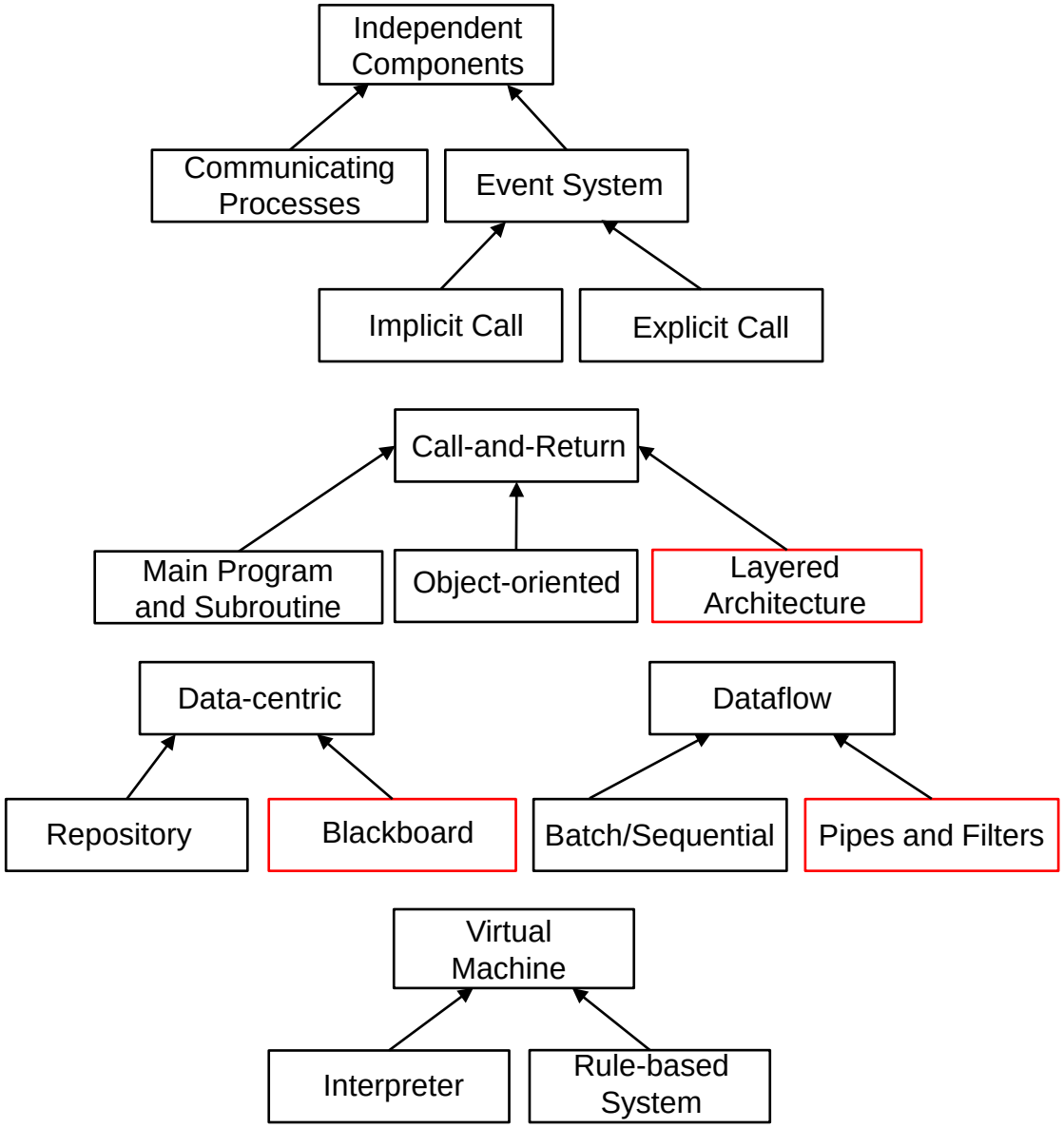
"Big data" is high-volume, -velocity and -variety information assets that demand cost-effective, innovative forms of information processing for enhanced insight and decision making *Gartner 2011*

– Mostly infrastructure-oriented
– Building applications is outside the style

Not a disadvantage, rather a matter of focus

– It can be difficult to separate streaming and batch data
  - Example: Orders in an online store (from 1000 a day to Millions a day …) - whether this is streaming or batch may depend on the application (real-time analytics)

Where can we place SOA, P2P and C/S architectures in this picture?



Conway's Law

Conway's law is an adage named after computer programmer Melvin Conway, who introduced the idea in 1968; it was first dubbed Conway's law by participants at the 1968 *National Symposium on Modular Programming*.

It states that organizations which design systems ... are constrained to produce designs which are copies of the communication structures of these organizations
—M. Conway

Conway, Melvin E. (April 1968)
"How do Committees Invent?", *Datamation* **14** (5): 28–31

Recommendations for Component Structures

- An architectural style should never depend on a specific product/tool!
- Component functionality is well-defined and implements information hiding and separation of concerns
- Data producing and data consuming components are clearly separated from each other
- Interfaces are well defined and “hide” change
  - Development teams can implement components independently of each other based on interface definitions
- Only a few types of interaction should be present in the system
- The same type of interaction should be implemented in the same way
- Relationship between development components and runtime components are understood (not nec. 1-1)
- Processes should be easily modifiable, if necessary at runtime
- Resource conflicts are recognized and their resolution is clearly defined

Working Questions

1. What do we understand by an architectural style?
2. Explain examples of architectural styles, their essential components, component technology, connectors used and constraints that have to be considered.
3. What the advantages and disadvantages of a given architectural style?
4. Many P2P architectures use a late binding topology. What quality attributes can require or prevent such a solution?
5. SOA includes dynamic service registry and discovery. Which quality attributes are affected positively or negatively?
6. How can a layer architecture implement the following tactics for modifiability: abstract common services, encapsulate, use an intermediary (reduce coupling)?

Summary

- 7 basic architectural styles that are widely applied and can be observed in many systems
- Any system has an architectural style
  - Remember “big ball of mud”
- Each style has advantages and disadvantages
- Applying more than a single architectural slide constitutes a risk due to increased complexity and potential constraint violations
- Select the simplest style based on your scenarios