UNIVERSITÄT DES SAARLANDES

30 YEARS DFKI Human Centric AI.

Architectural Pattern for Enterprise Application Integration (EAI)

Architectural Thinking for Intelligent Systems

Winter 2019/2020

Prof. Dr. habil.Jana Koehler

Agenda

- Concept and description of architectural pattern
- Enterprise Application Integration Patterns (EAI)
  1. ***File Transfer***

     Applications generate files for shared data to exchange
  2. ***Shared Database***

     Read and write applications to a common database
  3. ***Remote Procedure Invocation***

     Applications provide interfaces that can be accessed externally
  4. ***Messaging***

     Applications use a common messaging system for data exchange and retrieval

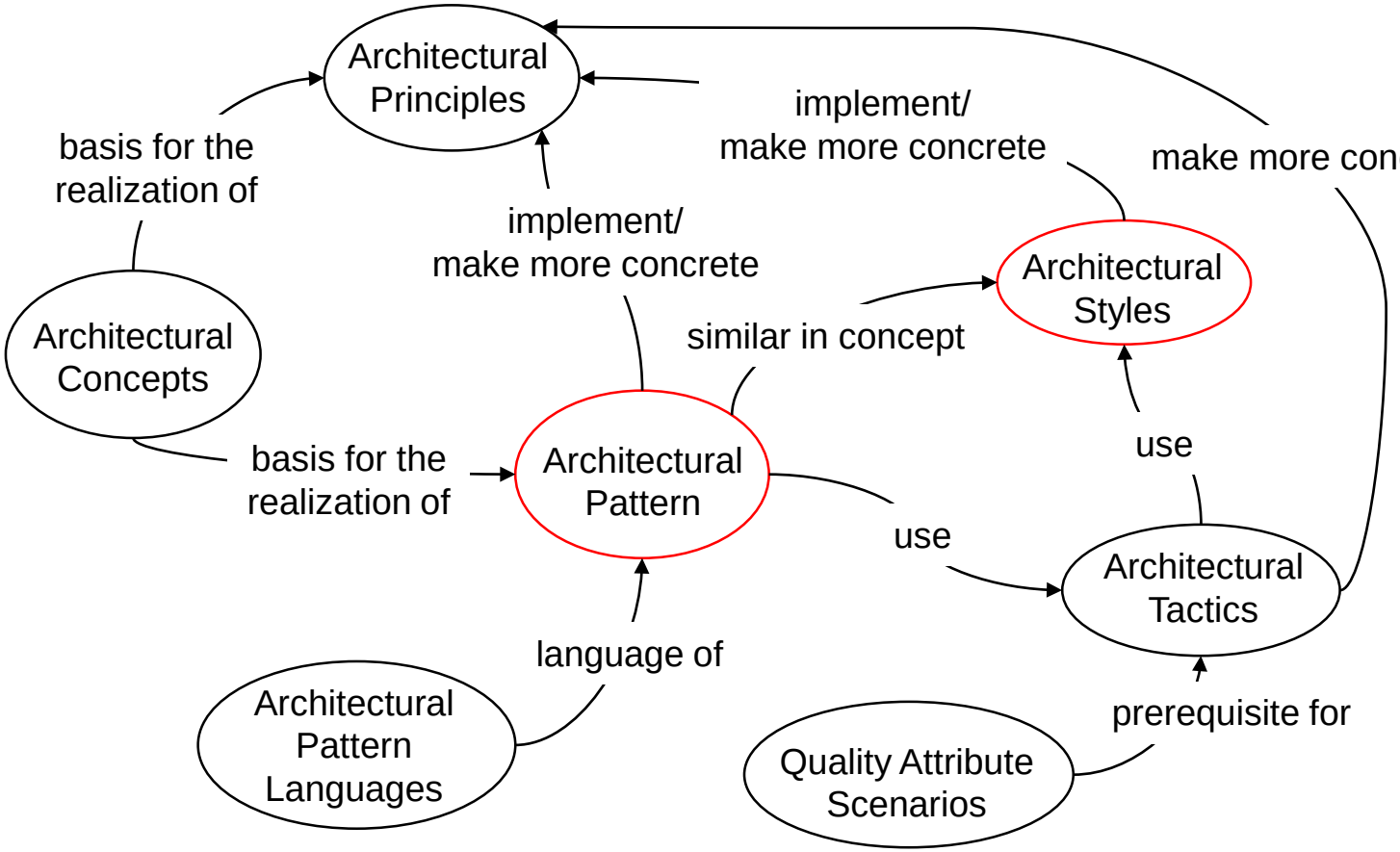
Tutorial Assignment 10:

- We take decisions about the application of specific EAI pattern in the architecture to integrate subsystems and components.

- We discuss the advantages and disadvantages of a specific pattern in the context of our system architecture and develop the rationale for the decisions.

Styles and Patterns (Revisited)

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



Vogel et al.

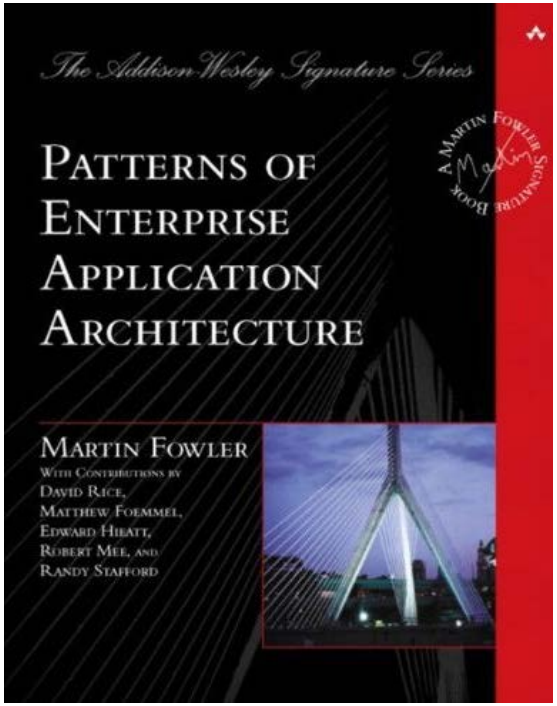
Architectural Pattern

- Vehicle to share practical experience
  - Are not invented, but discovered
- A pattern implements (several) tactics
- Is always a compromise between several quality attributes
- Task of the architect: instantiate the pattern
  - Apply current context and constraints
  - Carefully balance any compromise between quality requirements as implemented by the (instantiated) pattern

Description of a Pattern

- Context
  - Application situation that generates the problem
- Problem
  - the different forces acting in the problem
  - which problem variants can occur
  - quality attributes that must be fulfilled
- Solution
  - element types and their interaction
  - topology of the elements
  - constraints
  - degree to which quality attributes are fulfilled

A Seminal Book on Patterns



Patterns for the following key decisions the architect must take:

- Layering of an application
- Structuring of business logic
- Structuring of the (web-based) user interface
- Integration of a database
- Concurrency and transaction management
- Stateful vs. Stateless

Enterprise Integration Patterns (Hohpe et al.)

- *«Interesting applications rarely live in isolation.«*
- EAI patterns provide technology-independent solutions for the integration of systems and components
- Our definition from the beginning of this course:
  - "The software architecture of a system is the set of structures needed to reason about the system, which compromise software elements, **relations** among them and **properties** of both."

Integration Patterns

1. ***File Transfer***

   Applications generate files of shared data for exchange

2. ***Shared Database***

   Applications read and write from/to a common database

3. ***Remote Procedure Invocation***

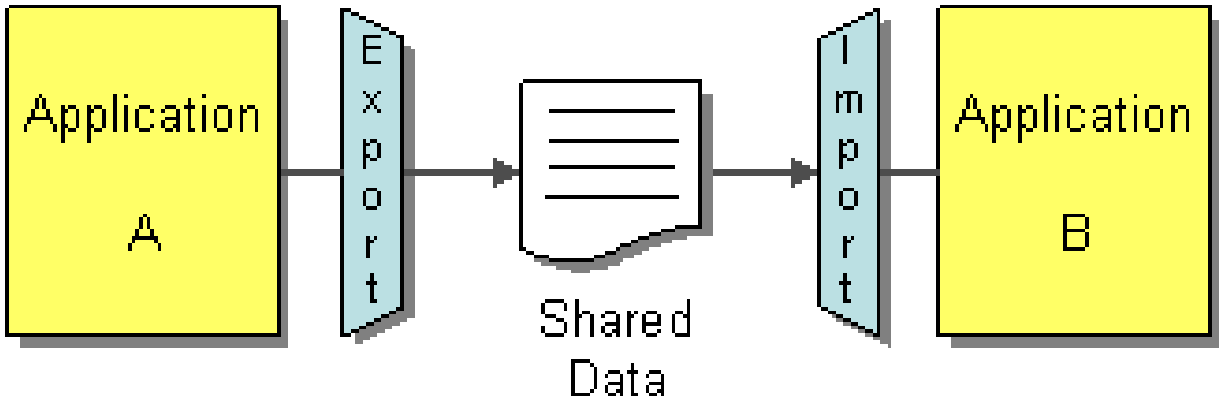
   Applications provide interfaces that can be accessed externally

4. ***Messaging***

   Applications use an underlying messaging system for data exchange and retrieval

1. File Transfer

- Definition of the file format and the time intervals in which the files are created and consumed



- Where to store files, how to manage them, how to deal with changes in the data format (i.e. file format)?

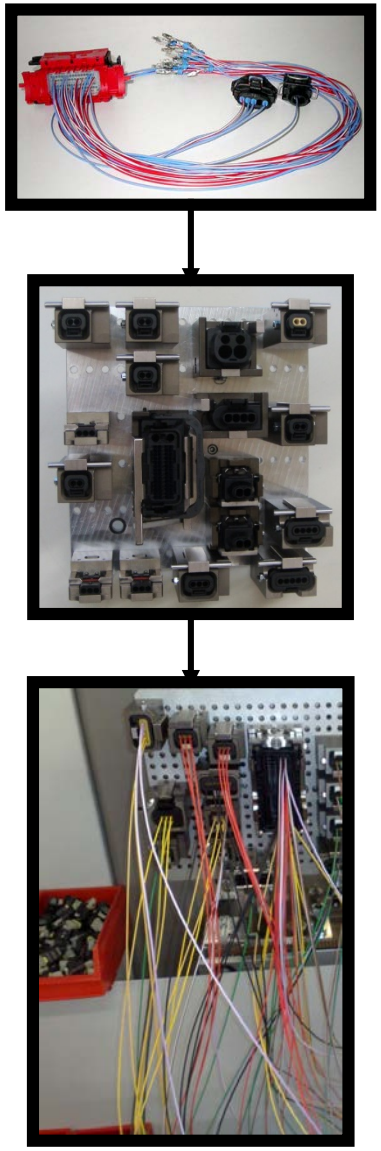
Advantages and Disadvantages of File Transfer

+ Files as a universal storage medium
+ Minimum hardware/software requirements
+ No knowledge of the application necessary for integration
+ No further dependencies between integrated applications

– Synchronization of data can be lost quickly when change happen frequently
– Unsuitable for very frequent exchange of small or very large amounts of data (file management problem)

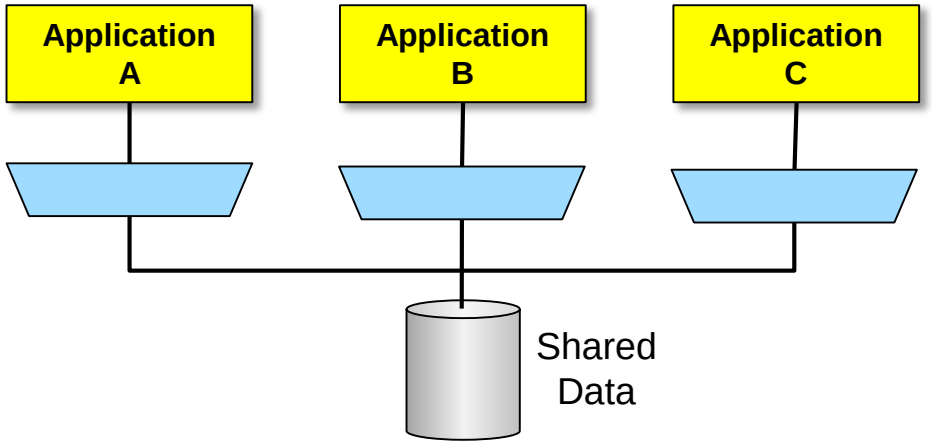
Example Cable Tree Wiring AI System

- VDA Vehicle Electric Container VEC
- Layout problem: placement of harnesses on palette
- Permutation problem: sequential ordering of plugging steps



2. Shared Database

- Suitable for frequent exchange of consistent data across multiple applications
- Definition and dissemination of uniform data formats



- Management of data exchange via database transactions ensured
  - Focus on DB schema definition / domain model

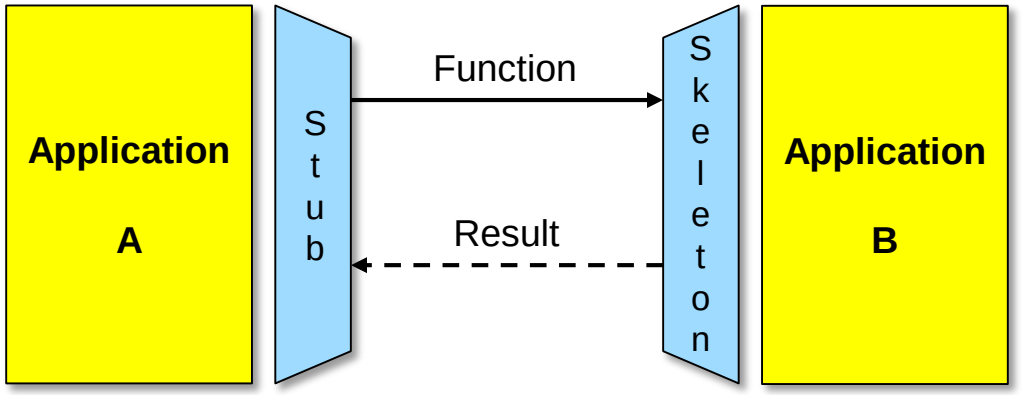
Advantages and Disadvantages of Shared Database

+ Support of database standards in all development environments
+ Synchronization of data guaranteed by DB transactions

– Applications must adapt to data formats
  – Requires adapters, transformations
– Agreement of all applications on a uniform data format
– External applications often only work with their own formats (need for additional adapters)
– Database as a potential single point of failure for all applications
  – performance and scalability requirements

3. Remote Procedure Invocation

- Integration of functionality, not just data
- Applications work together by calling each other



- Applications as components, data access encapsulated and available via functional interfaces
  - CORBA, REST calls

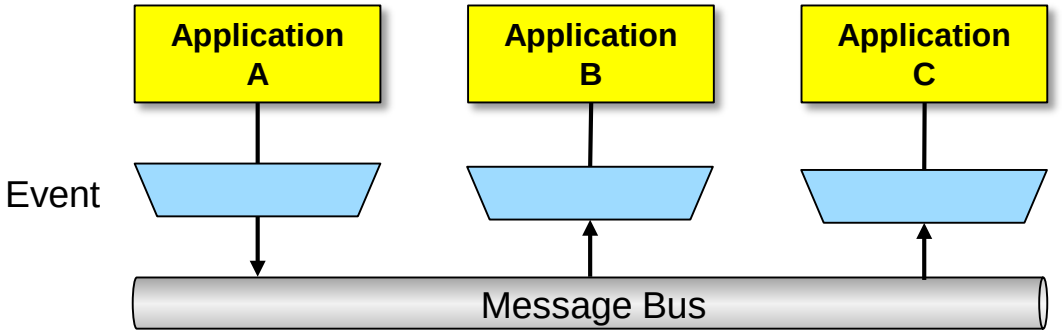
Advantages and Disadvantages of Remote Procedure Invocation

+ Different interfaces can be provided
+ Applications can change internal data formats (information hiding)
+ Simple development concept ("procedure call")

– Coordinating and changing interfaces across multiple application boundaries can be difficult
– Maintainance effort for legacy interfaces
– Performance and reliability problems of remote calls ("remote" <> "local")
– Close coupling of applications ("growing knot")

4. Messaging

- Loose coupling of systems by frequent exchange of small data packets
- Different communication styles
  - Synchronous, Asynchronous, Publish-Subscribe



- Loose coupling (including functions) despite diverse integration possibilities using different exchange formats

Advantages and Disadvantages of Messaging

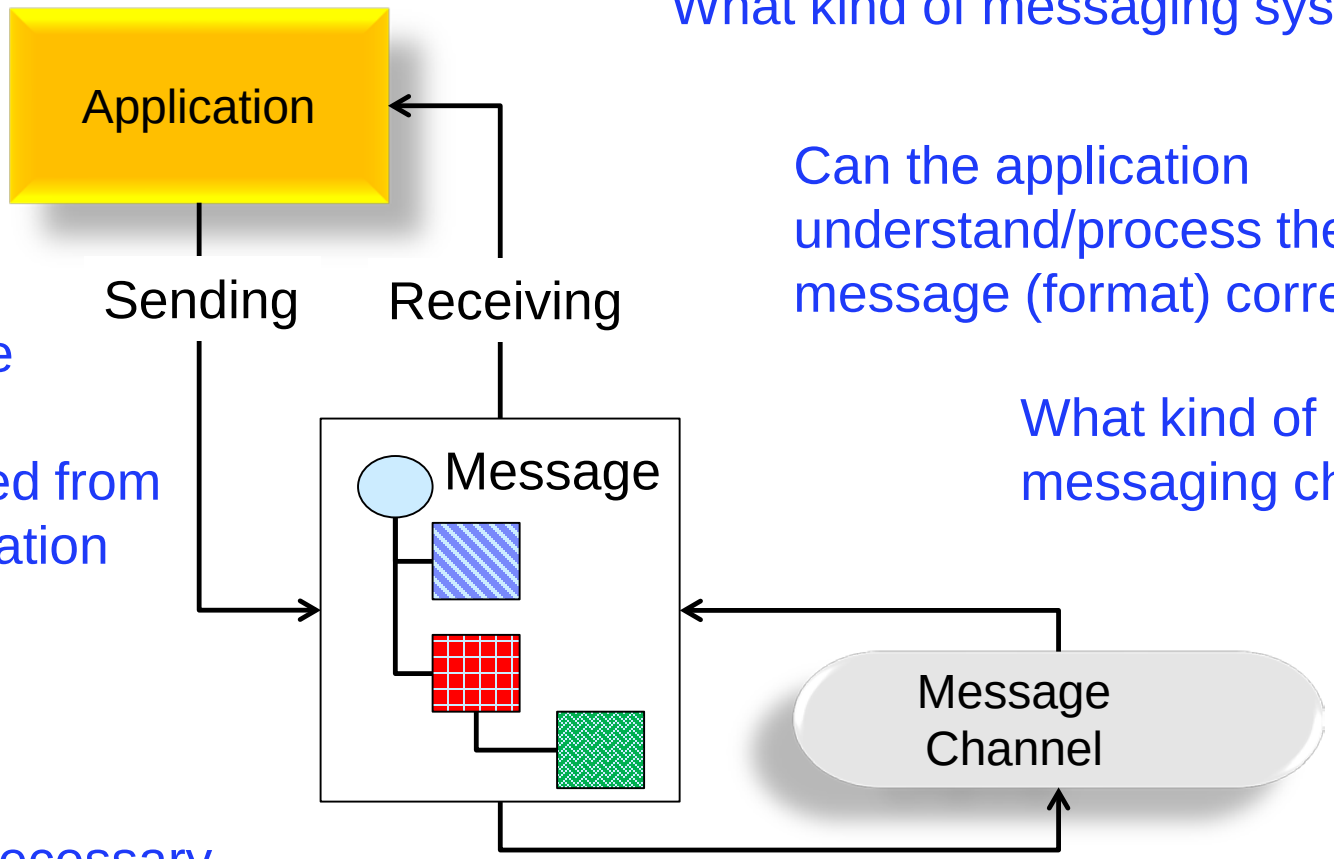
+ Most flexible integration pattern
+ Supported by a variety of technology standards
+ Transformation and management of data happens within the messaging middleware and not in the application

– Asynchronous design of application functions not always easy
– Testing and debugging more difficult
– Flexibility vs. simplicity/complexity

Messaging decisions and patterns

How are applications connected with the messaging system?

What kind of messaging system?



Can the application understand/process the message (format) correctly?

How is the message constructed from the application data?

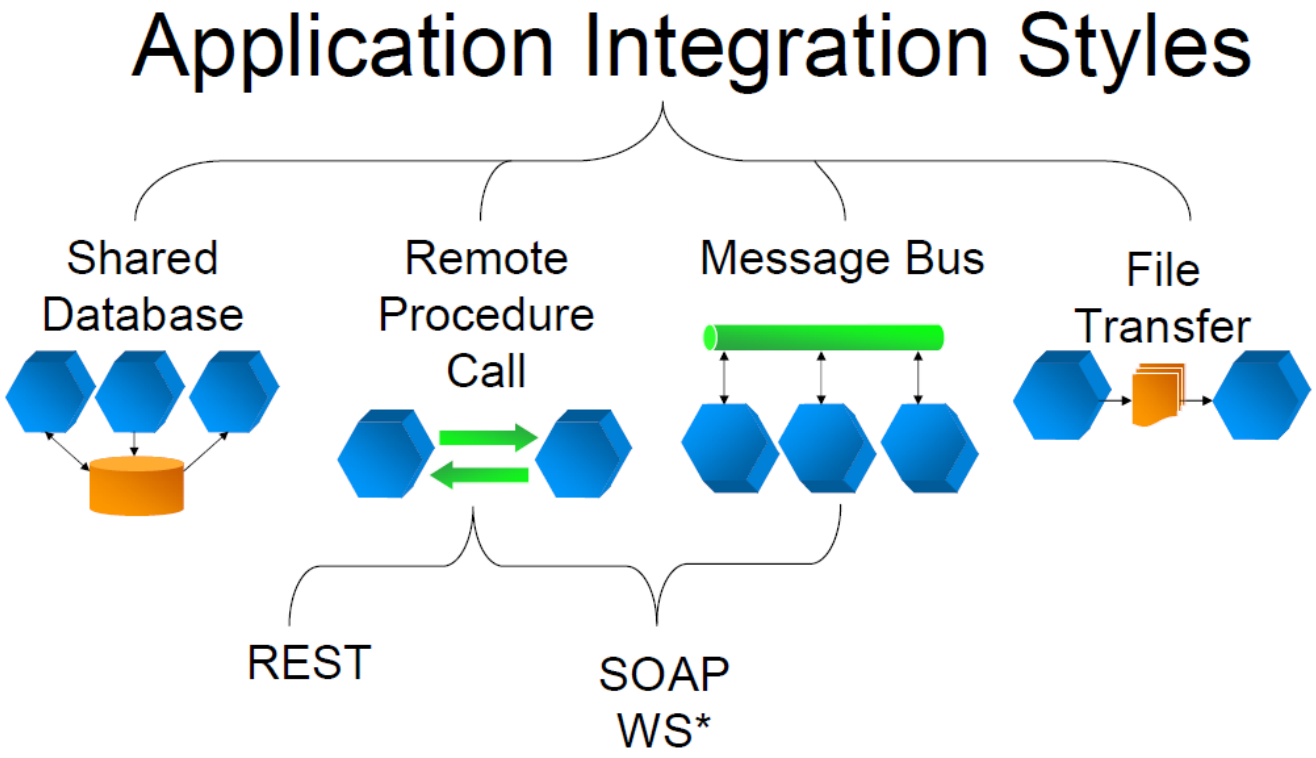
What kind of messaging channel?

How are necessary transformations carried out?

How does the message reach the correct receiver(s)?

EAI and Architectural Decisions

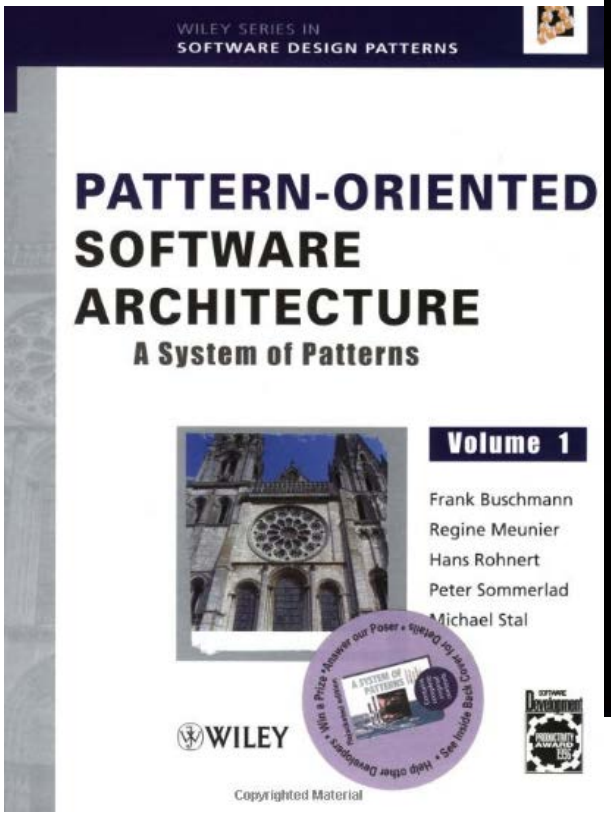
- *Pautasso, Zimmermann, Leymann: RESTful Web Services vs. "Big" Web Services: Making the Right Architectural Decision*

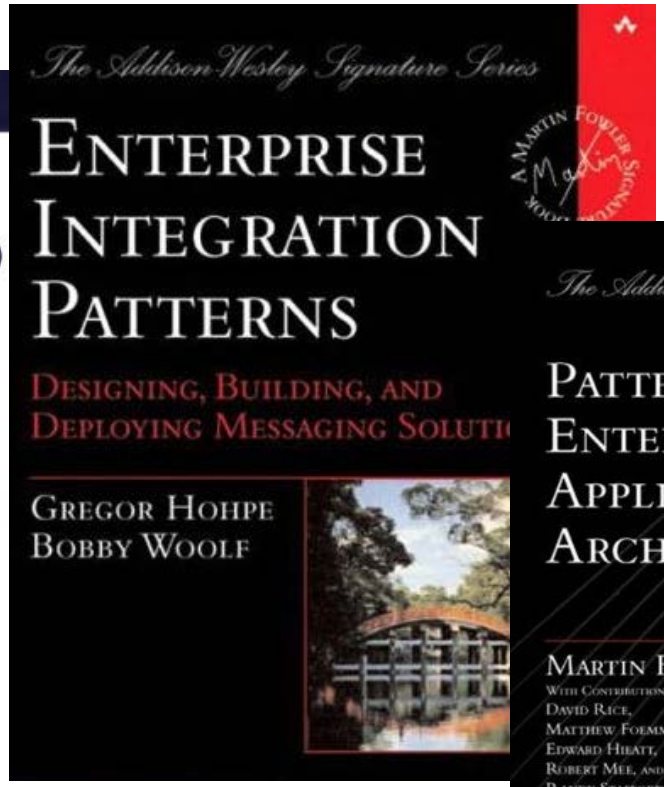


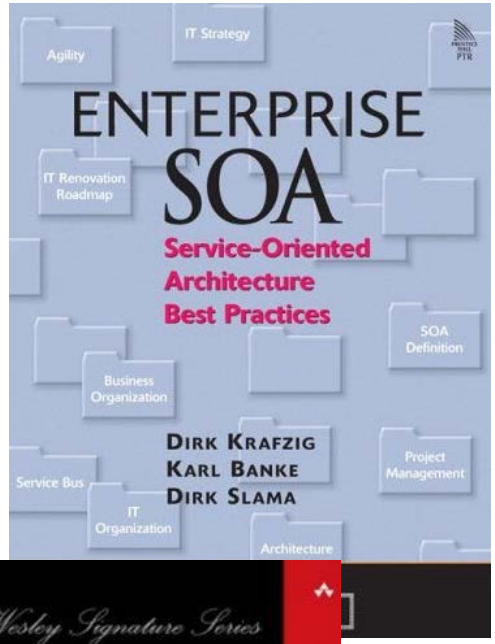
http://www.jopera.org/files/www2008-restws-pautasso-zimmermann-leymann.pdf

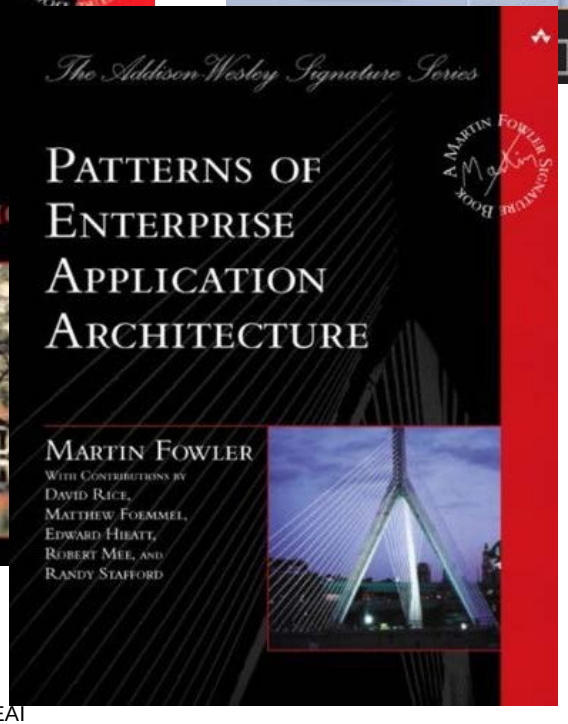
More about Patterns

- http://eaipatterns.com/
- http://www.martinfowler.com/eaaCatalog/









Summary

- 4 widely used basic integration patterns that are universally applicable
- Each pattern has advantages and disadvantages
  - Measurable quality attributes in the form of scenarios help in decisions
- System architecture should be open for the evolution of the integration solution
- Keep as simple as possible

Working Questions

1. What do we understand by an architectural pattern?
2. How are architectural patterns described?
3. How do architectural patterns differ from design patterns, e.g. OO design pattern?
4. Explain examples of integration patterns and discuss their advantages and disadvantages.
5. Discuss the difference between tactics, styles, and patterns.
6. Which quality attributes influence the selection of a synchronous or asynchronous communication?
7. Which principles are influenced by the selection of an EAI pattern?