Enterprise Application Integration

Herbert Kuchen

University of Münster

Mo 14:15 - 15:45, Fr 10:15 - 11:45, Leo 18

Summer 2013

Contents

- Assignments: 4 (solved in small groups)

Literature

- G. Hohpe, B. Woolf: Enterprise Integration Patterns, Addison Wesley, 2004.
- D.S. Linthicum: Next Generation Application Integration: From Simple Information to Web Services, Addison-Wesley, 2003.
- S. Conrad, W. Hasselbring, A. Koschel, R. Tritsch: Enterprise Application Integration: Grundlagen - Konzepte - Entwurfsmuster - Praxisbeispiele, Spektrum Akad. Verlag, 2005. (in German)
- M.B. Juric, K. Pant: Business Process Driven SOA Using BPMN and BPEL, Packt Publishing, 2008.
- E. Roman et. al.: Mastering Enterprise JavaBeans 3.0, Wiley 2006, pdf available for free.
- W. Eberling, J. Lessner: Enterprise JavaBeans 3.1, Hanser, 2011. (in German)

Preconditions and Goals of this Course

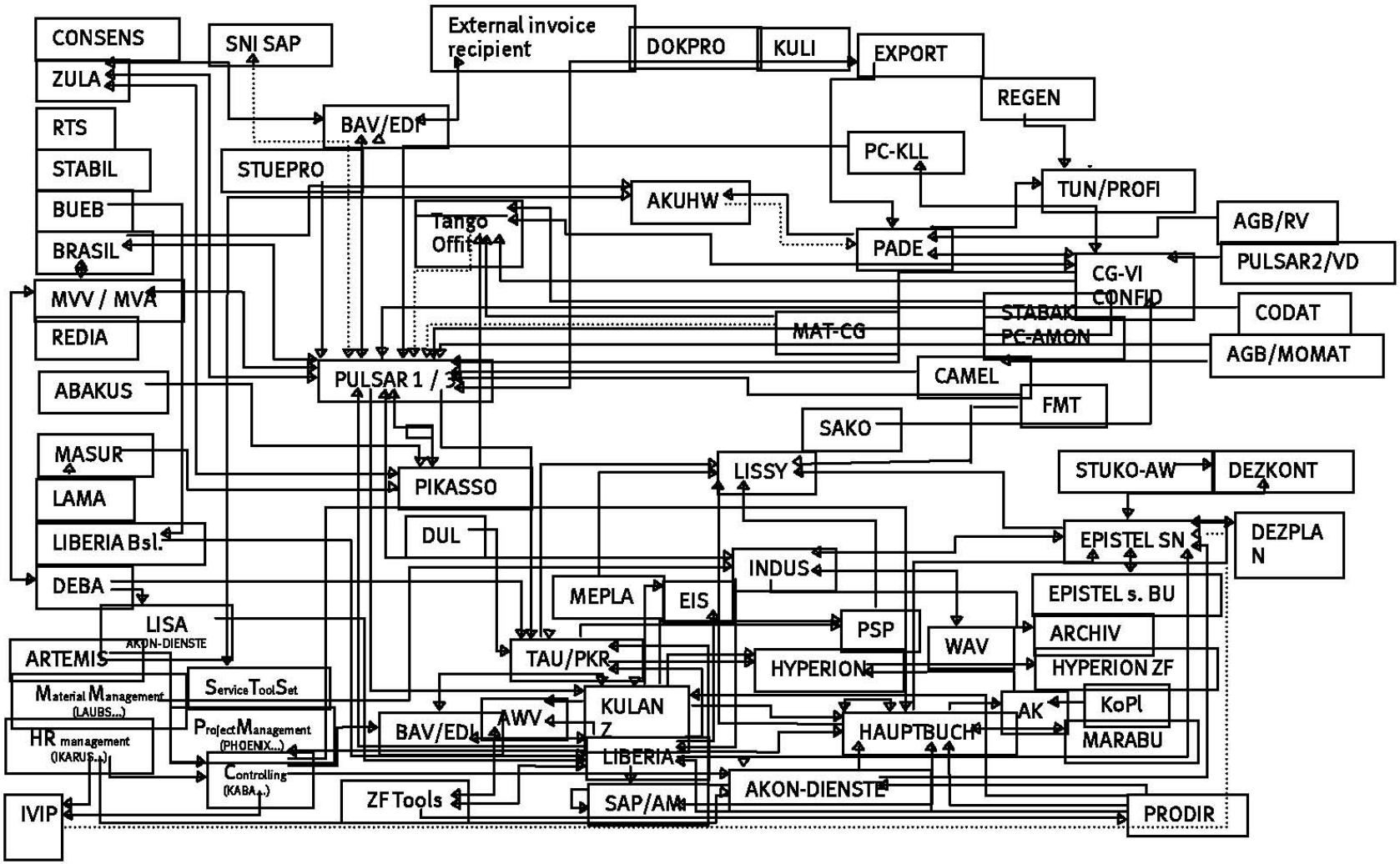
- Preconditions: skills in programming and software engineering
- Goals:
  - convey knowledge and practical experience w.r.t. the intra- and inter-enterprise integration of information systems
  - strengthen the ability to develop software in a team

Part I

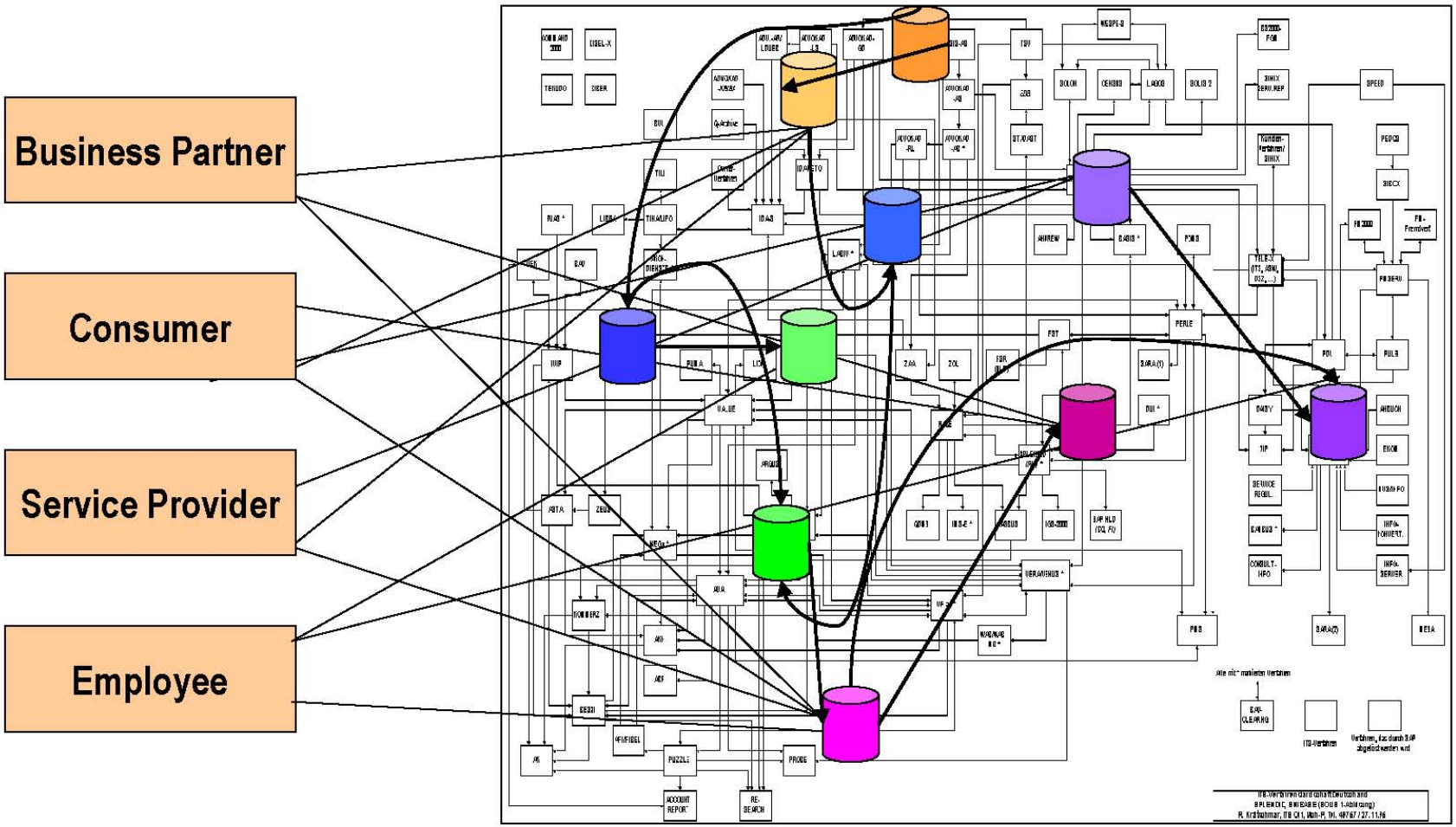
EAI: Foundations, Concepts, and Architectures

Outline

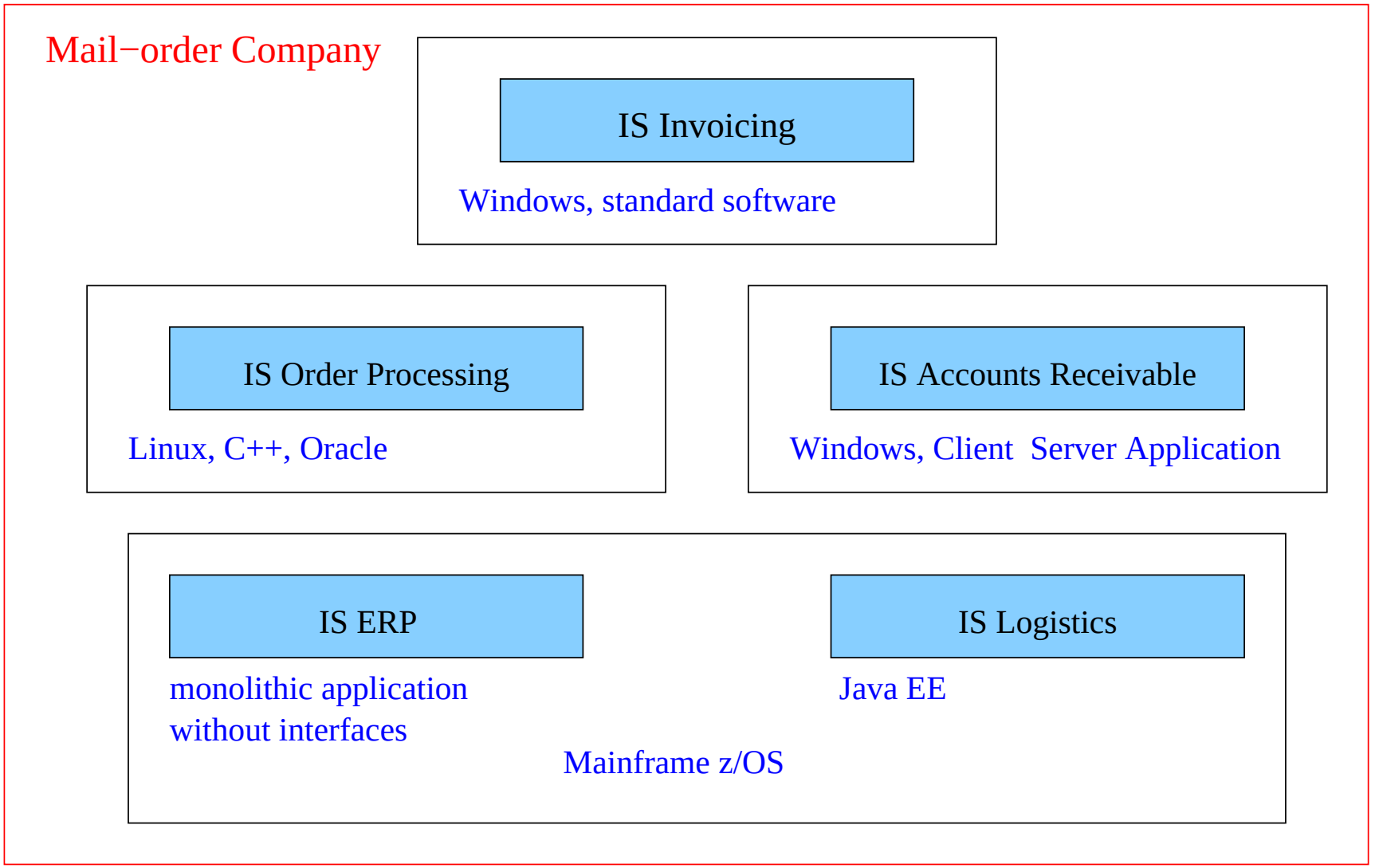
What is the Problem?



What else is the Problem?



Example: Mail-order Company



Example: Mail-order Company (continued)

- Problem:
  - each department uses own specific IS
  - order and client data are stored redundantly
  - information exchange impeded by different platforms, formats, technologies, and architectures
- Goal: integrated support of business processes

Why Enterprise Application Integration (EAI)?

- improve the efficiency and effectiveness of business processes e.g. by eliminating media breaks
- create new business processes by connecting existing information systems
- implement mergers and acquisitions
- save the investment in existing systems

Outline

Definition: EAI

- EAI addresses the integration of information systems inside of and across enterprises
- EAI comprises
  - methods
  - standards
  - technologies
  - architecture concepts

Heterogeneity

- Technical Heterogeneity
  - different DBMS, hardware, operating systems, network components, ...
- Data Heterogeneity ($\rightarrow$ Course "Data Integration")
  - different styles of data modeling (e.g. relational vs. OO)
  - schema conflicts due to differing use of modeling constructs (e.g. inheritance vs. delegation)
  - data conflicts due to e.g. inconsistent naming, types, data range, scaling, ...
  - semantic conflicts; e.g. "employee" includes or excludes manager
- Organizational Heterogeneity
  - ISs maintained by different organizational units

Example: Data Heterogeneity

homonym	(legal) process	(business) process
synonym	employee	staff
data types	int	String
scaling	1.75 m	175 cm
precision	0.5276	0.53
integrity constraints	salary < 8000	salary < 9000

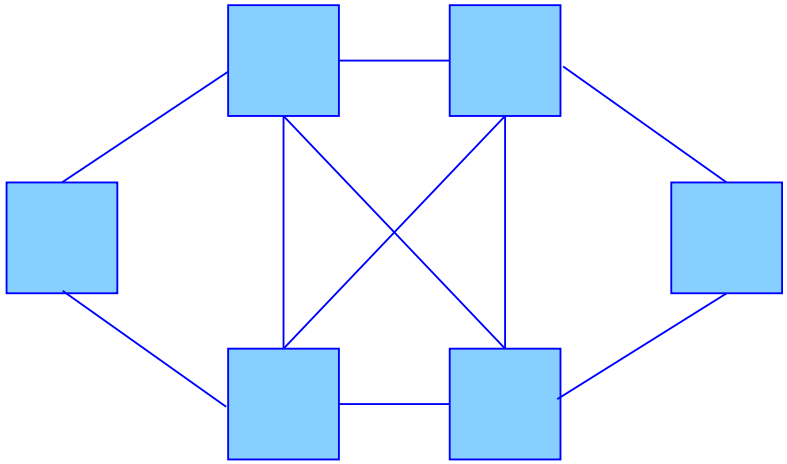
- different spelling: Weseler Straße, Weselerstr.
- moreover: typos, obsolete data, ...

Outline

Integration Architectures

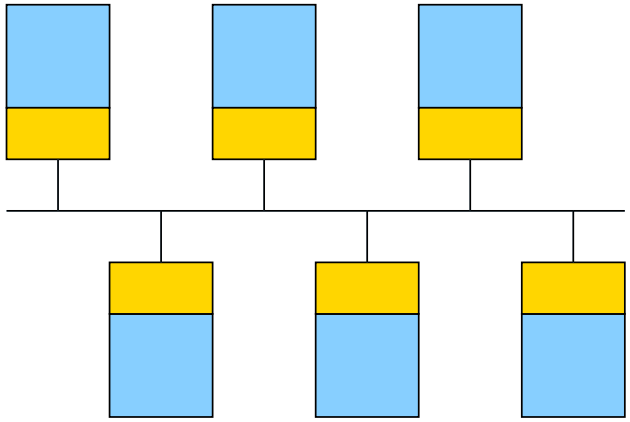
- Integration Topologies
  - point to point
  - bus
  - hub and spokes
- Integration Layers
  - data layer
  - business logic layer
  - presentation layer

Point-to-Point Integration



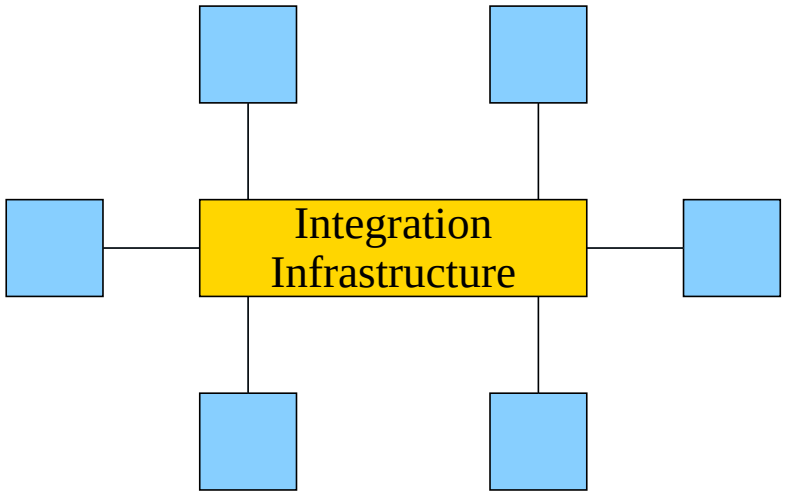
- demand-driven connection by specific interfaces
- Advantage:
  - easy to implement in small scenarios
- Disadvantages:
  - tight coupling of systems
  - up to $n * (n-1)/2$ interfaces
  - hence expensive maintenance

Bus Topology



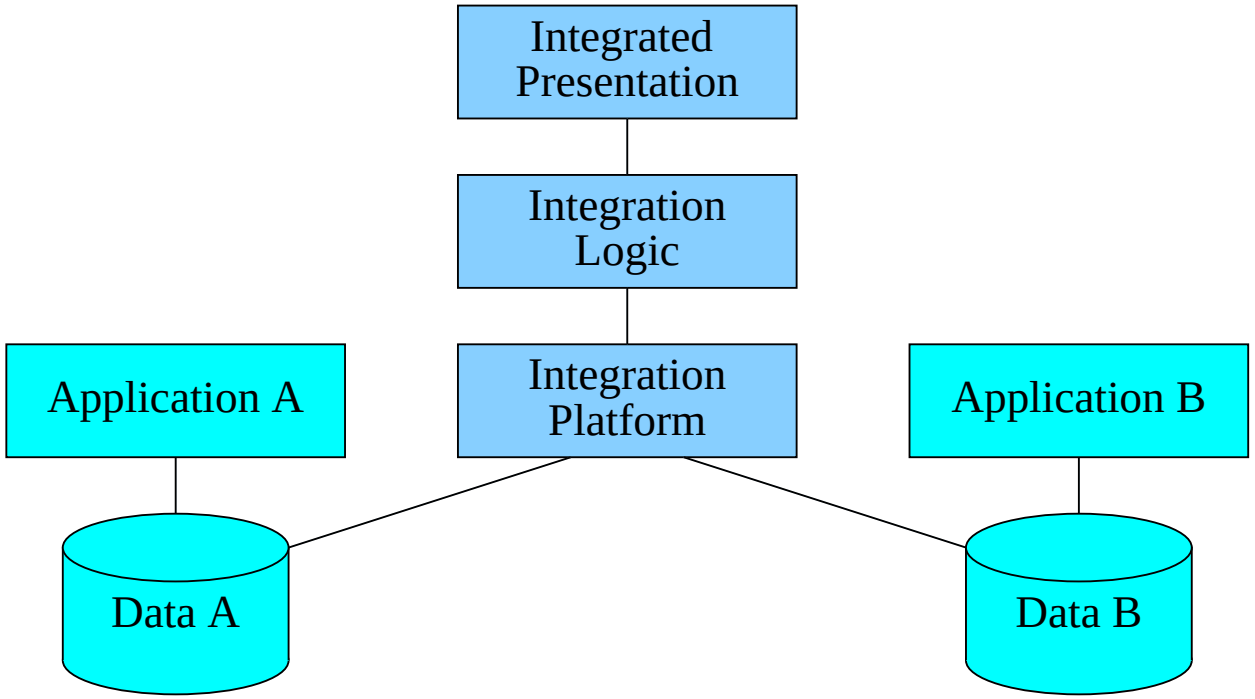
- Advantages:
  - few interfaces
  - good scalability
  - no single point of failure
- Disadvantages:
  - redundant local infrastructure functionality
  - coordination overhead

Hub and Spokes Topology



- systems connected via central hub; standardized interfaces
- Advantages:
  - few interfaces
  - easy integration of additional systems
- Disadvantages:
  - initial overhead for installation of central integration infrastructure
  - hub is potential bottleneck and single point of failure

Integration on the Data Layer

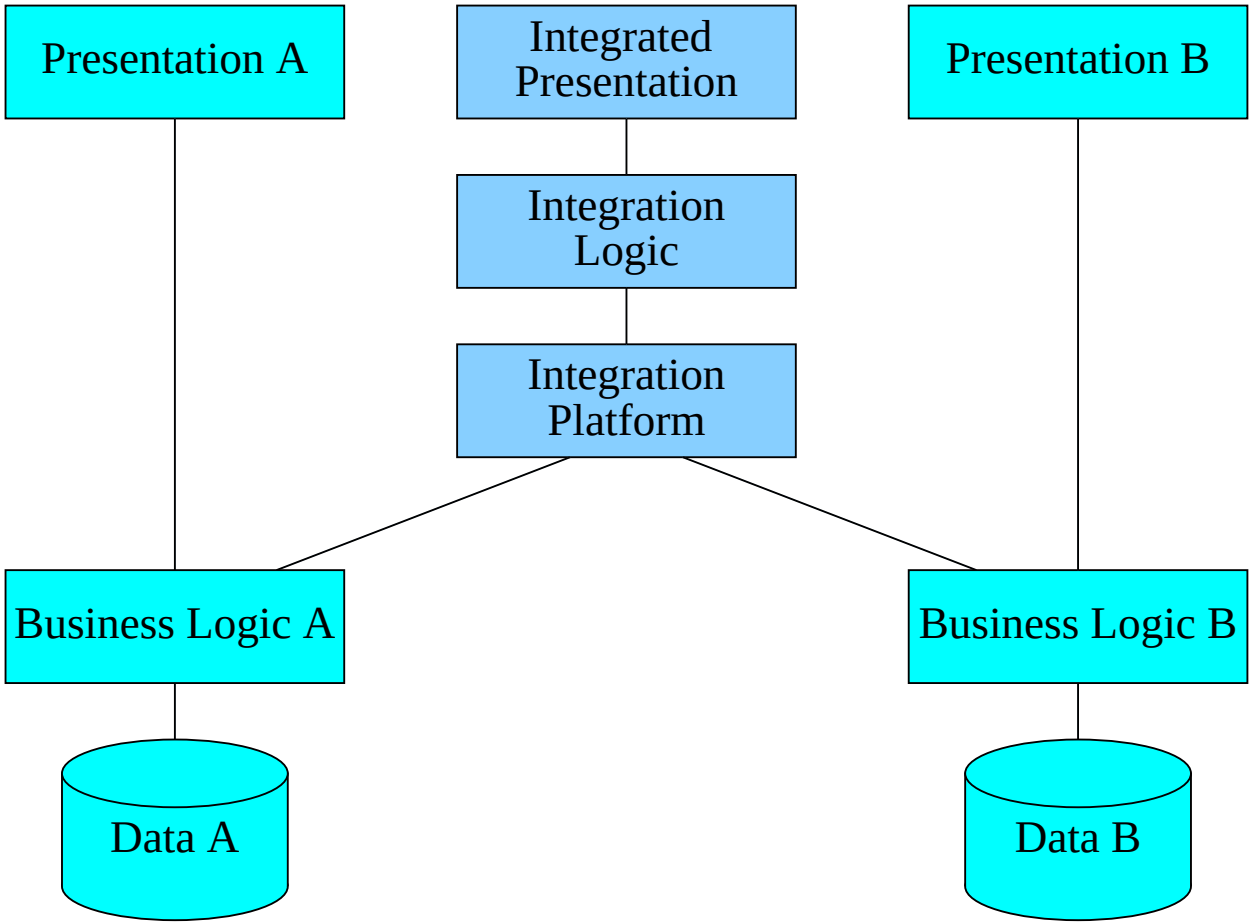


- direct access to data bases of existing applications
- for details see course “Data Integration”
- existing business logic and presentation logic ignored

Integration on the Data Layer: Properties

- Advantages:
  - easy to implement
  - even possible without source code of existing applications
- Disadvantages:
  - sematic problems (e.g. integrity constraints guaranteed by applications may be violated by write accesses)
  - existing business logic not used
  - changes of the data model require updates of transformation rules

Integration on the Business Logic Layer

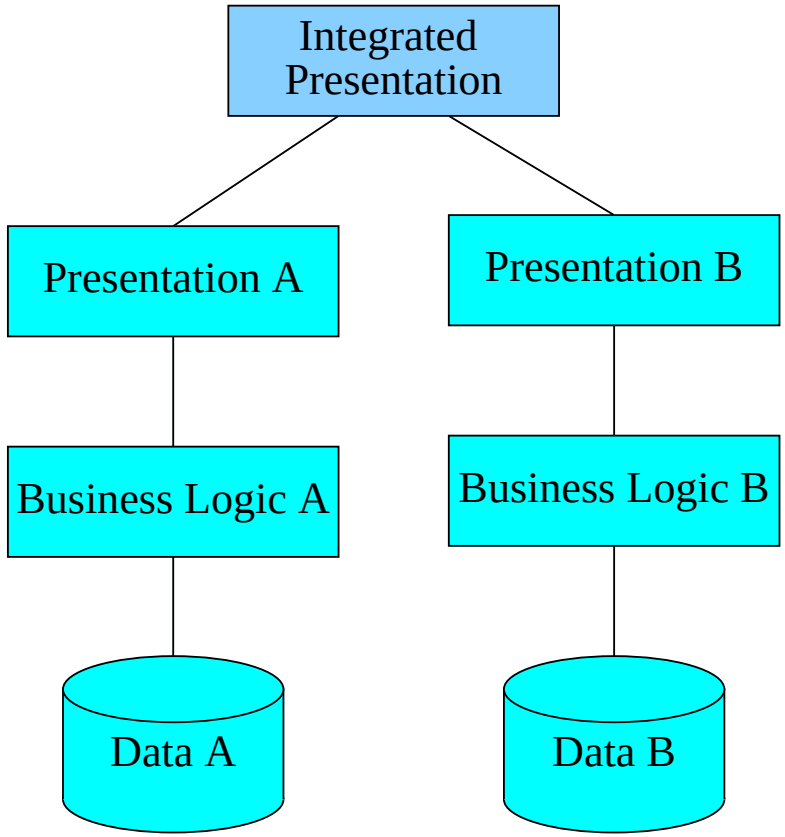


- semantically richest and most flexible integration
- access e.g. via RPC, RMI, CORBA, EJB, DCOM, MOM

Integration on the Business Logic Layer: Properties

- Advantages:
  - semantically richest way of integration
  - business logic of existing applications used
  - no problems with bypassed integrity constraints
- Disadvantages:
  - possibly additional interfaces and adapters have to be implemented
  - implementation potentially complex ($\rightarrow$ increased risk)

Integration on the Presentation Layer



- based on screen scraping, WSRP, HttpUnit, ...
- connection to application by parsing the corresponding screen and by simulating a user dialog ($\rightarrow$ tools)

Integration on the Presentation Layer: Properties

- used for integrating web applications (HTML,→ DOM)
- used for integrating (monolithic) legacy systems without interfaces
- mainly used, if other approaches fail
- Advantages:
  - easy to implement with appropriate tools
  - applicable even if application provides no interfaces
  - only applicable approach in difficult integration scenarios
- Disadvantages:
  - no real integration of data and functionality
  - bad performance and low flexibility

Outline

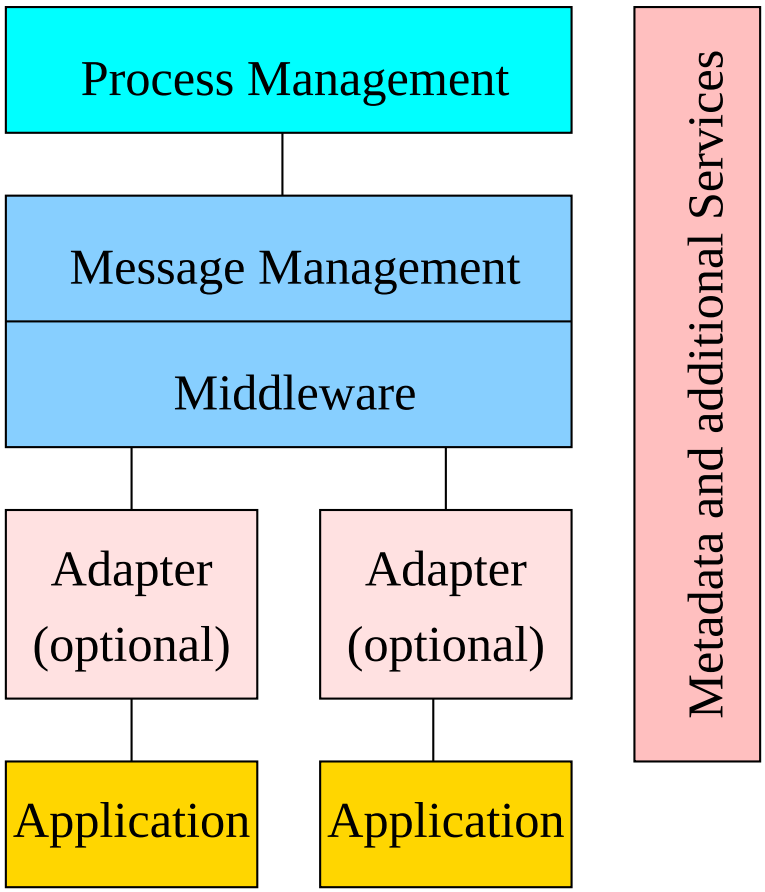
Elements of an EAI-Infrastructure

process modeling, monitoring, benchmarking

transformation

synchronization

transaction control

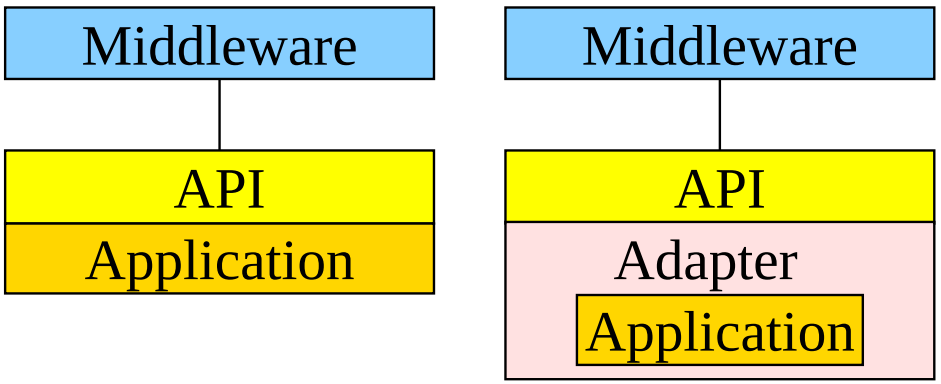


system and service management

identity management

administration

Integration without and with Adapter



- adapter performs simple transformations

Middleware / Message Management

- infrastructure for communication between applications
- bridges heterogeneity
- ensures reliable communication
- optional: rule based message distribution
- supports transactions
- provides notification services
- data transformation (if not handled by adapter)

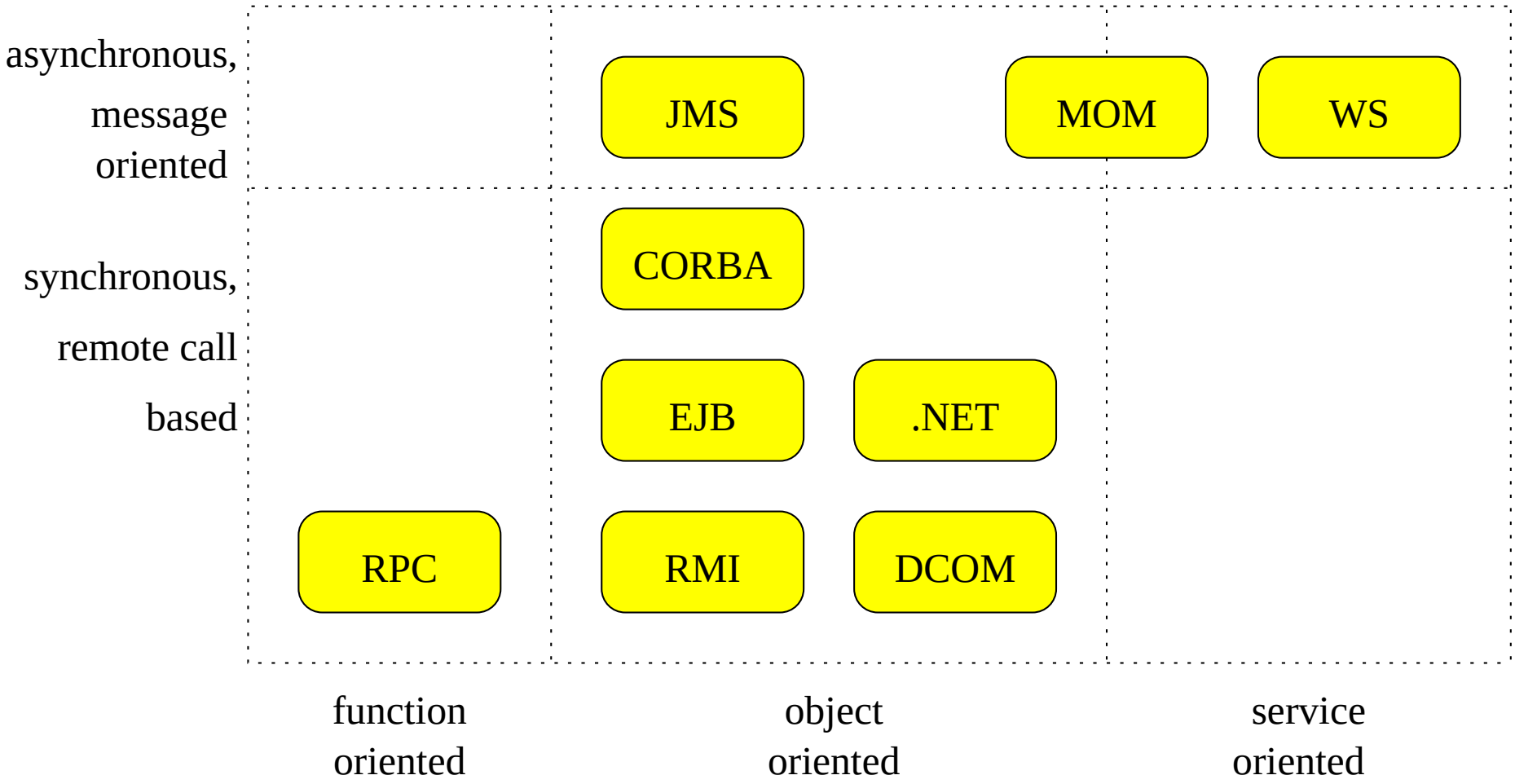
Communication Modes

- synchronous:
  - sender sends message to receiver and waits for answer
  - needed in time critical applications
  - sender blocks, if receiver is unavailable or message is lost
- asynchronous:
  - sender sends message, but does not wait for an answer
  - often: messages stored until they are deliverable
  - this requires a complex infrastructure
  - can simulate synchronous communication

Integration Paradigms

- function oriented
  - communication via procedure or functions calls
  - lack of encapsulation ($\rightarrow$ global variables) causes tight coupling
- object oriented
  - communication via method calls
  - encapsulation due to defined interfaces
- service oriented
  - service typically coarse grained with interface description
  - services can (sometimes) be found in registries
  - access often message based; loose coupling

Regulation Framework: Integration Technologies



- combining technologies allows to combine their advantages

Process Management

- enables the interaction of applications and hence process integration
- modeling, execution, controlling and monitoring of business processes
- independent of the implementation of each application
- each application provides steps of a superior business process

Metadata and Additional Services

- maintenance of central information which is relevant for the entire EAI-solution
- maintenance of services, resources, users, roles
- authentification and authorization (identity management)
- monitoring of services and interactions
- administration of the infrastructure

Outline

Remote Produre Call and Remote Method Call

- Remote Procedure Call (RPC)
  - protocol for calling remote procedures and functions
  - hides transmission details (marshalling, demarshalling)
  - synchronous
- Remote Method Invocation (RMI)
  - similar to RPC
  - allows to call remote methods

Disadvantages of RPC / RMI

- not platform independent (RPC) and not programming language independent (RPC/RMI)
- sender blocked, if receiver unavailable or message lost
- inflexible, since addressing is encoded in the source code of the sender (no runtime addressing)

Common Object Request Broker Architectue (CORBA)

- component architecture designed by OMG
- provides applications as distributed objects
- communication based on RMI
- platform-independent interface description based on Interface Definition Language (IDL)
- hardware platforms and network structure transparent
- repository for searching objects
- only for OO languages
- limited interoperabilty of different CORBA implementations

Enterprise JavaBeans

- framework for developing object-oriented, component-based distributed Java applications
- platform independent
- communication with remote components based on RMI (or JMS)
- persistence automatically handled
- automatic transaction control
- authorization and authentification

Properties of EJBs

- Advantages:
  - platform-independent, object-oriented
  - container provides basic services such as persitence, transactions, security
  - asnchronous communication available via Java Messaging Service (JMS)
- Disadvantages:
  - not programming-language independent (Java only)
  - integration of applications at runtime difficult

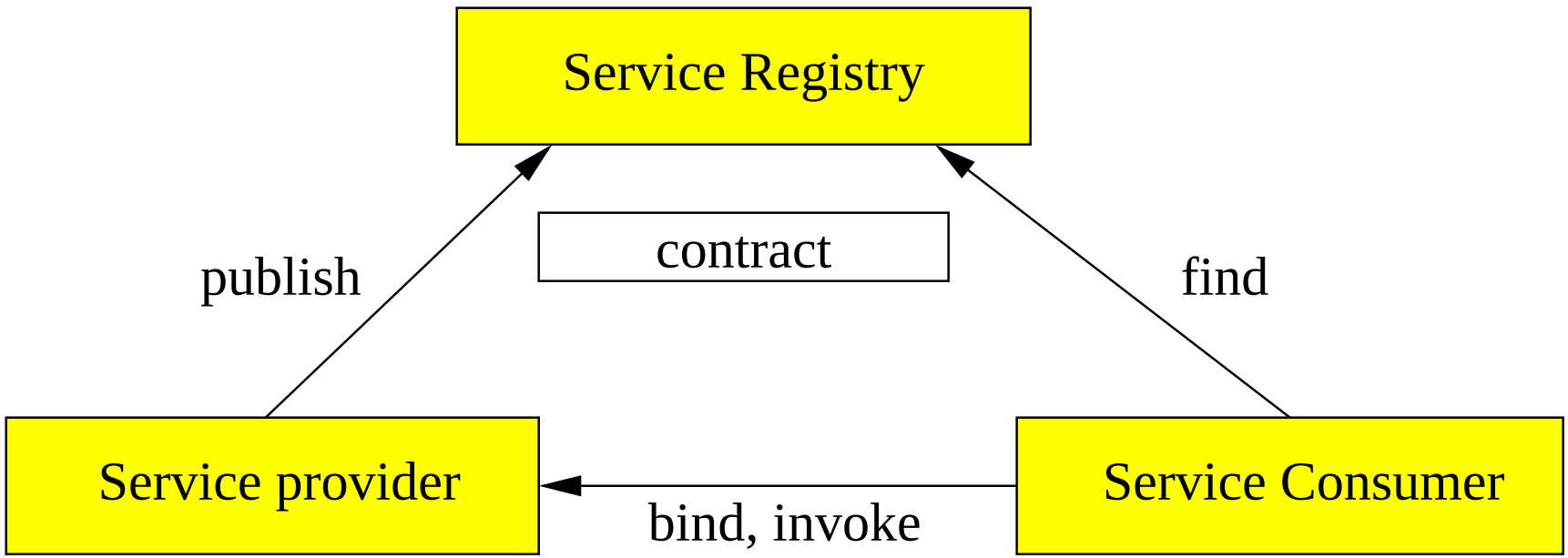
Message Oriented Middleware (MOM)

- communication via (asynchronous) messages
- messages stored in queues until they can be delivered to their destination(s)
- multicasting (to several destinations) possible
- independent of platforms and technologies
- messages contain metadata and data

Service Oriented Architectures (SOA)

- SOA is an integration concept rather than a technology
- a software is decomposed into a set of services
- a service is a modular functionality with an interface
- services represent reusable building blocks of business processes
- new services can be created by combining existing ones
  ($\rightarrow$ BPEL)
- optional: services published in registries
- there they can be found and integrated in advance
  or even at runtime

SOA Interactions



Web Service

- loosely coupled, reusable service
- interface and service description independent of implementation
- description in XML format WSDL
- communication based on XML format SOAP
  (on top of HTTP, FTP, SMTP, JMS, . . .)
- optional: registry and discovery based on XML format UDDI

Properties of Web Services

- Advantages:
  - platform, programming language, and protocol independent
  - interoperable due to open standards (SOAP,WSDL,UDDI)
  - even possible (but not recommended):
    addressing and integration at runtime
  - support synchronous and asynchronous communication
- Disadvantages:
  - performance loss due to XML encoding
  - for security, additional standards have to be used
    (e.g. WS-Security)

Business Process Execution Language (BPEL)

- XML-based programming language for combining web services to business processes
- includes exception and event handling ($\rightarrow$ compensation)
- a BPEL program requires a BPEL runtime environment for execution