IDC Austria: SOA Conference

From Decoupled Services to Agile Business Applications

Alexander Schatten

Inst. of Software Technology and Interactive Systems

TU-Wien

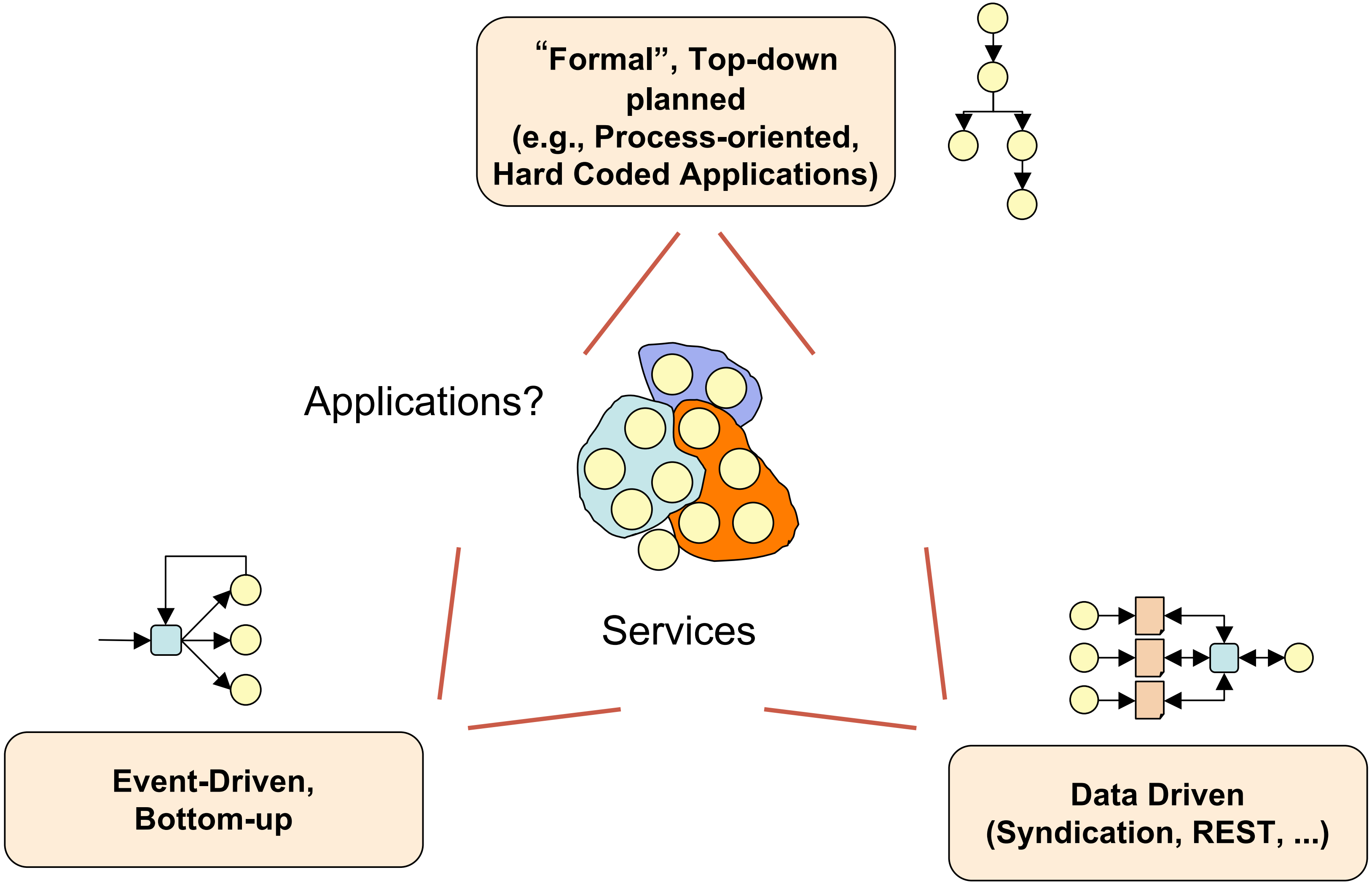
www.schatten.info

The Motto for this Talk ?

"Reusing other people's code would prove that I don't care about my work. I would no more reuse code than Ernest Hemingway would have reused other authors' paragraphs.”

One of Apples important developers in the late 80s

The Motto for this Talk !
Building Agile Applications from Reusable Services



"Formal", Top-down planned (e.g., Process-oriented, Hard Coded Applications)

Applications?

Services

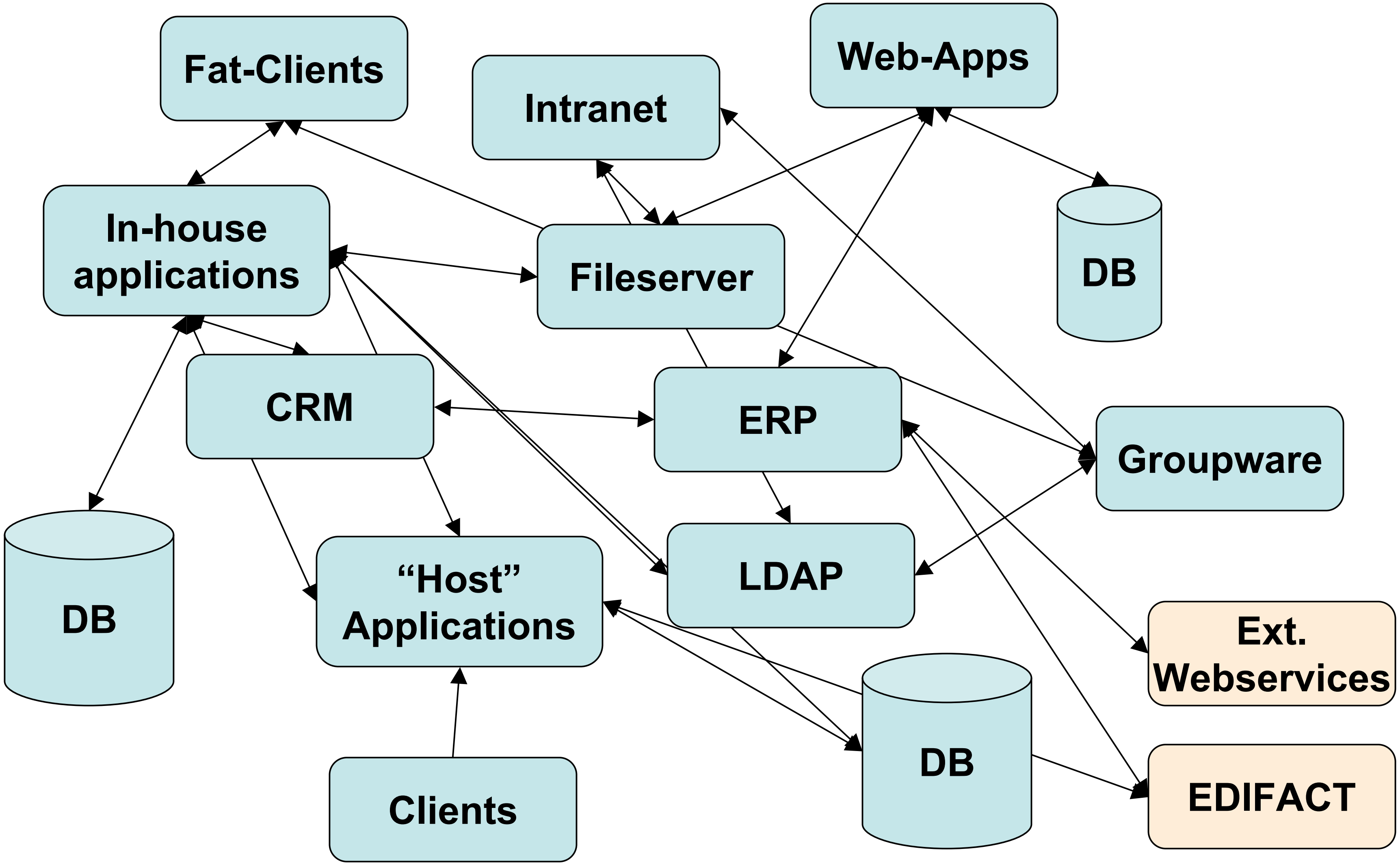
Event-Driven, Bottom-up

Data Driven (Syndication, REST, ...)

Agenda

- Status Quo? Basic Considerations about Services
- Services and Protocols
- Having Services... What next? Views on Aggregation
- Integration Middleware
- **Styles of Aggregation**
  - "Formal"
  - Event-Driven
  - Data-Driven
- Conclusion

Status Quo?


Fat-Clients
Intranet
Web-Apps
In-house applications
Fileserver
DB
CRM
ERP
Groupware
DB
"Host" Applications
LDAP
Ext. Webservices
DB
Clients
EDIFACT

Motivation

- Different Systems should be able to cooperate with less effort
  - Within company (system)
  - Between companies
- Clean Architecture
- Less Integration Effort
- *Agile* IT Services desired
- Monitoring and Management of IT and Processes should be transparent

Agility?

"Agility is the successful exploration of competitive bases (speed, fexibility, innovation proactivity, quality and profitability) through the **integration of reconfigurable resources** and **best practices** in a knowledge-rich environment to provide customer-driven products and services in a fast changing market environment."

Yusuf et.al., Agile manufacturing: The drivers, concepts and attributes,
Int. J. Production Economics 62 (1999) 33–43

Going Service-Oriented...

Service?

- Rather course grained **Component**
  - exposes **functionality** through
  - **interoperable interface**
  - which is defined in a **platform neutral** way
- Accessible via Network
- Endpoint of a connection
- Service is ideally stateless
- Service Infrastructure allows
  - Loose coupled interaction of services
  - Aggregation, orchestration and choreography
  - Several “vertical” features (security, service level agreements, management, ...)

Stateful and Stateless Services

- Stateless Services
  - Each request is generic
  - Maximum Decoupling possible
  - High Reliability (not depending on one specific instance)
- Stateful Services
  - Often necessary
  - Keep Session between requests
  - Tighter Coupling
  - *Actually typically a part of a service aggregation!*

SOA?

- Modularity
- Loose Coupling
- Separation of Concerns
- Composition
- Independence
- Interoperability

„Computer Science is entering a new generation. The previous generation was based on **abstraction from hardware**. The emerging generation comes from **abstracting from software** and sees all resources as **services** in a SOA“, Michael Brodie et.al.

Service Identification

- Top-Down
  - Based on Processes
  - Based on Business-Goals
- Bottom-Up
  - Derived from existing services
  - Derived from existing applications/infrastructure
- Probably a mixed-approach

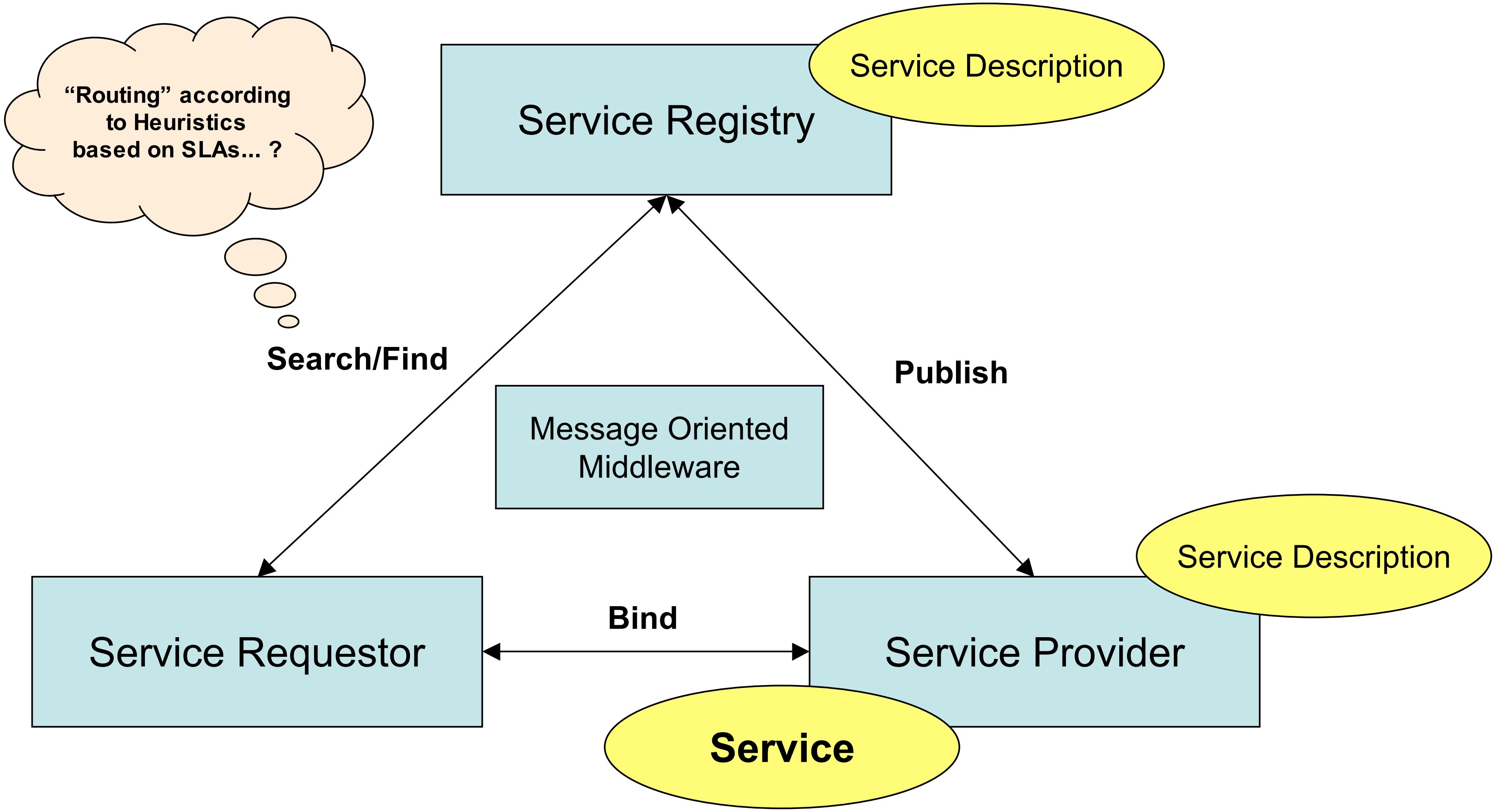
Loose Coupling

- **Reuse** of Functionality
- **Reduce dependencies** between systems
- Enhance Robustness
- Enhance Flexibility & Increase Interoperability through standard (XML) interfaces
- A “good service” is used by multiple clients

- Can be mediated by **Message Oriented Middleware**

However, loose coupling should *not* mean, that the behaviour of the system is sloppy and unclear or cannot be monitored and controled!

Decoupling with “Find-Bind-Invoke” Paradigm



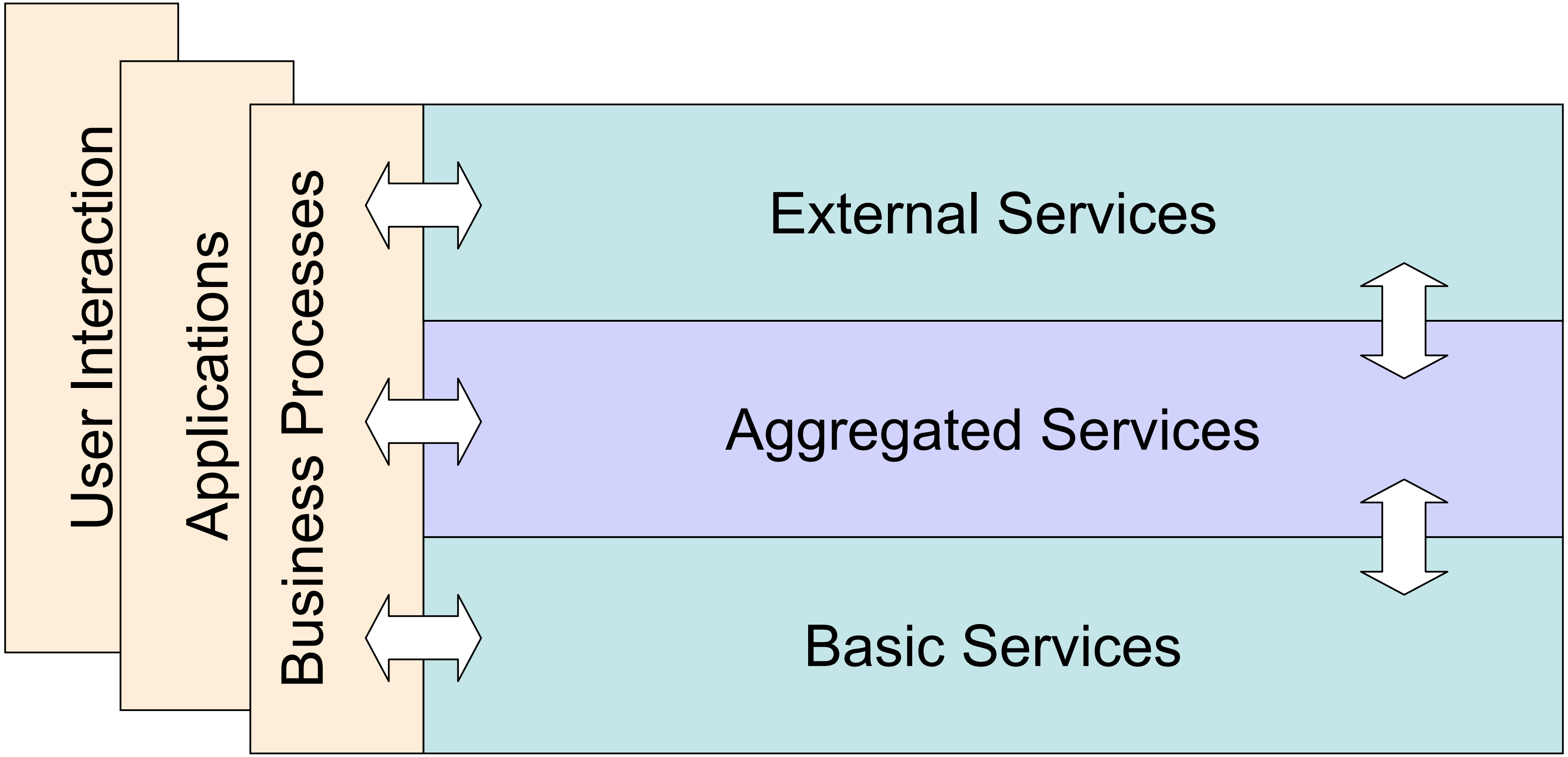
“Routing” according to Heuristics based on SLAs… ?

Having Services... What next?

- O.k. now we have a bunch of “perfect” services
- Example: **typical larger financial service provider** has **300-700 business functions** and probably nearly as many IT services ...

- However, we are actually not interested in decoupled services, but in executable business processes (workflows) and applications, or interoperation with other service units...
- Hence **recoupling** of services, i.e., aggregation, composition is often required

SOA Layers


User Interaction
Applications
Business Processes
External Services
Aggregated Services
Basic Services

Aggregation?

- **Direct connection** between services/components (scripting, developing applications with **direct** service invocation...)
- Connection via Registry (**Find-Bind-Invoke**)
- Connection via **“aggregation languages/standards”**e.g.
  - process-based
  - Orchestration
  - Choreography
  - Application oriented
- **“Loose connection”**... (over MOM, event driven, ...)

Formal vs. “Informal” and Proprietary Service-Protocols

(what are we dealing with?)

WS-* Stack

- “Complete” implementation of the *separation of concerns* idea
- **SOAP:** Message Format
- **WSDL:** Service Description
- **WS-Policy:** framework for non-functional requirements
  - WS-Security-Policy
- **WS-Addressing:** universal datastructure for endpoint addressing, list of header elements for service endpoints
- **WS-Reliable Messaging**
- **WS-Security** (XML-Encryption, XML-Signature)
- **WS-Trust**
- **WS-Transaction**
- **WS-Notification**
- **WS-BPEL:** Orchestration
- **WS-CDL:** Choreography
- ...

REST

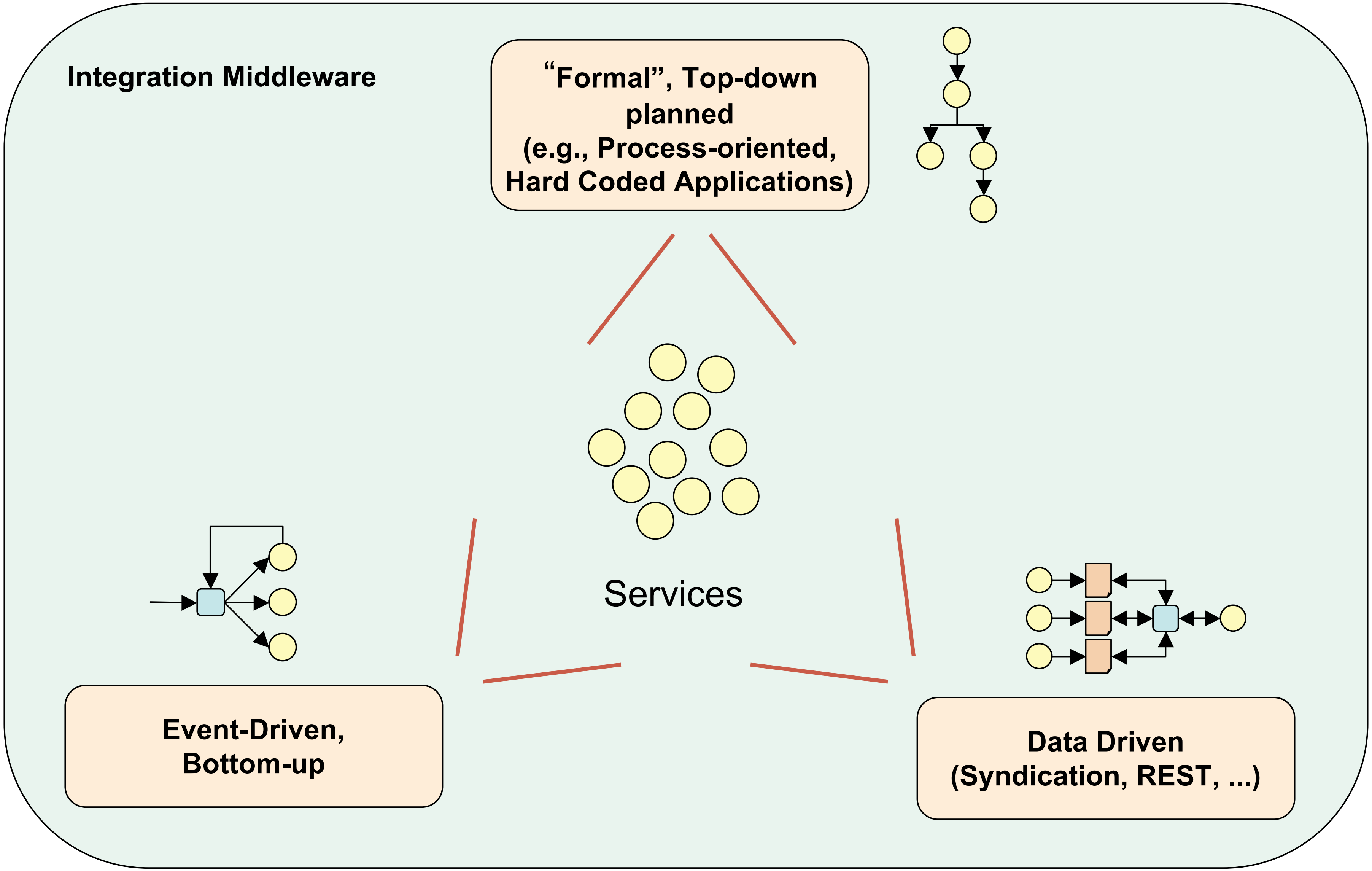
- **Representational State Transfer**
- Resources, no services
- Resources have representations (e.g. XML, HTML, gif, PDF, ...)
- **“Web-Style” Applications** (Tim Bray)
- No specification on transfer format (html, jpg, gif, **XML**)
- Based on http (get, post, put, delete)
- Everything (resource) is an URI
  - http GET mydomain.com/user/32213
  - http GET mydomain.com/article/a6557448x
  - http PUT mydomain.com/order
  - http DELETE mydomain.com/article/b443235
- Client/Server
- Stateless
- Cacheable, highly Scalable
- Beispiele: Google Base, del.icio.us

Proprietary Protocols

- EDIFACT
- Corba
- Binary Protocols
- ...

Views on Aggregation

Views on Aggregation



Integration Middleware

“Formal”, Top-down planned
(e.g., Process-oriented, Hard Coded Applications)

Services

Event-Driven, Bottom-up

Data Driven
(Syndication, REST, ...)

Integration Middleware and Standards

Enterprise Service Bus?
Example: Apache Service Mix

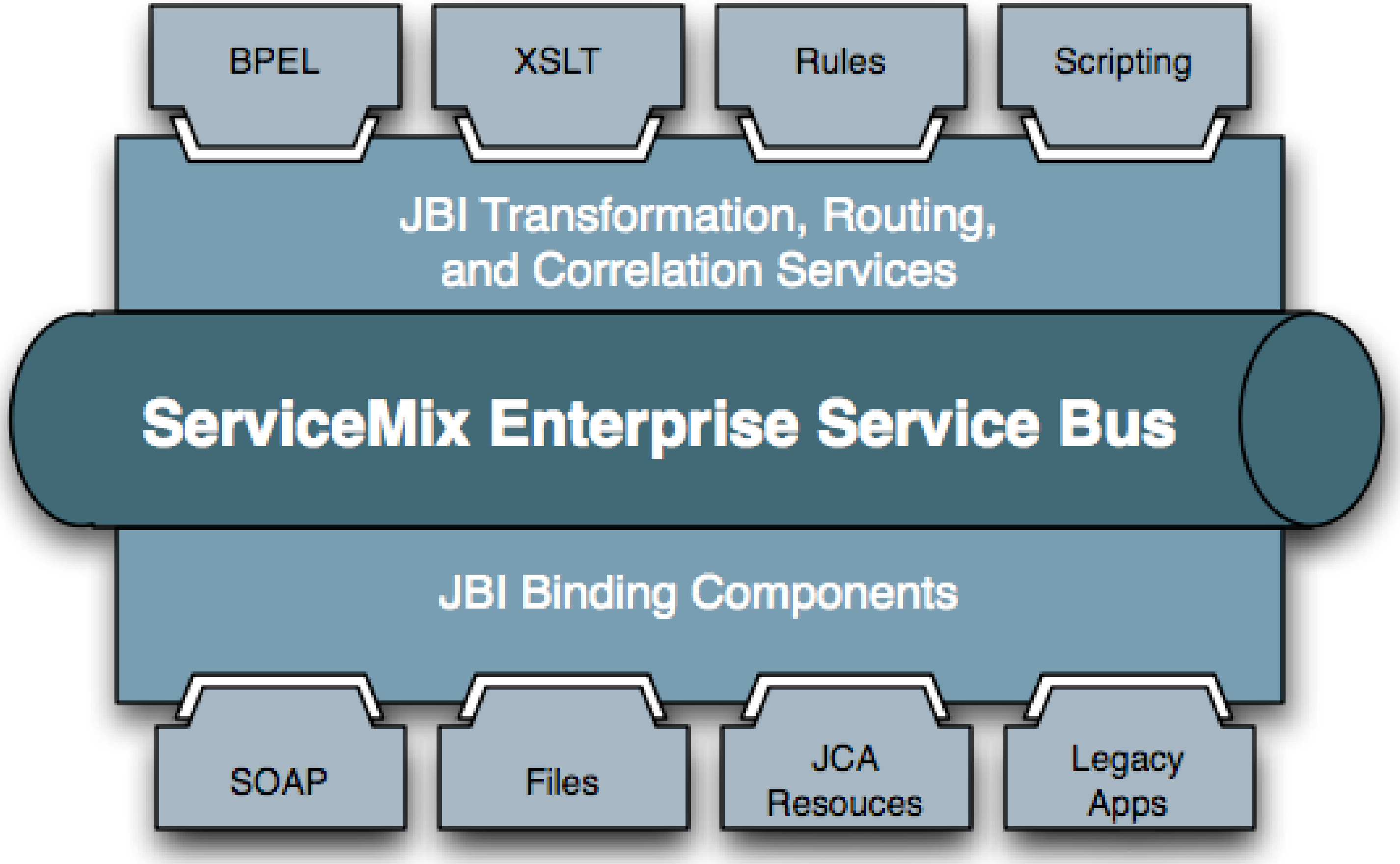

BPEL
XSLT
Rules
Scripting
JBI Transformation, Routing, and Correlation Services
ServiceMix Enterprise Service Bus
JBI Binding Components
SOAP
Files
JCA Resouces
Legacy Apps

Figure taken from Apache Service Mix documentation

JBI Standard

- Java Business Integration
- Container for integration components and services
- WSDL-Based Messaging Model
- Normalized Message Service and Router
- Components
  - Binding Components (e.g., JMS, Jabber, REST, ...)
  - Service Engines (BPEL, Camel, Quartz, Drools, ...)
- JMX Based Administration (Admin Tools)

- **JBI Components independent of ESB Implementation**

Aggregation using “Formal Planning”, Standards

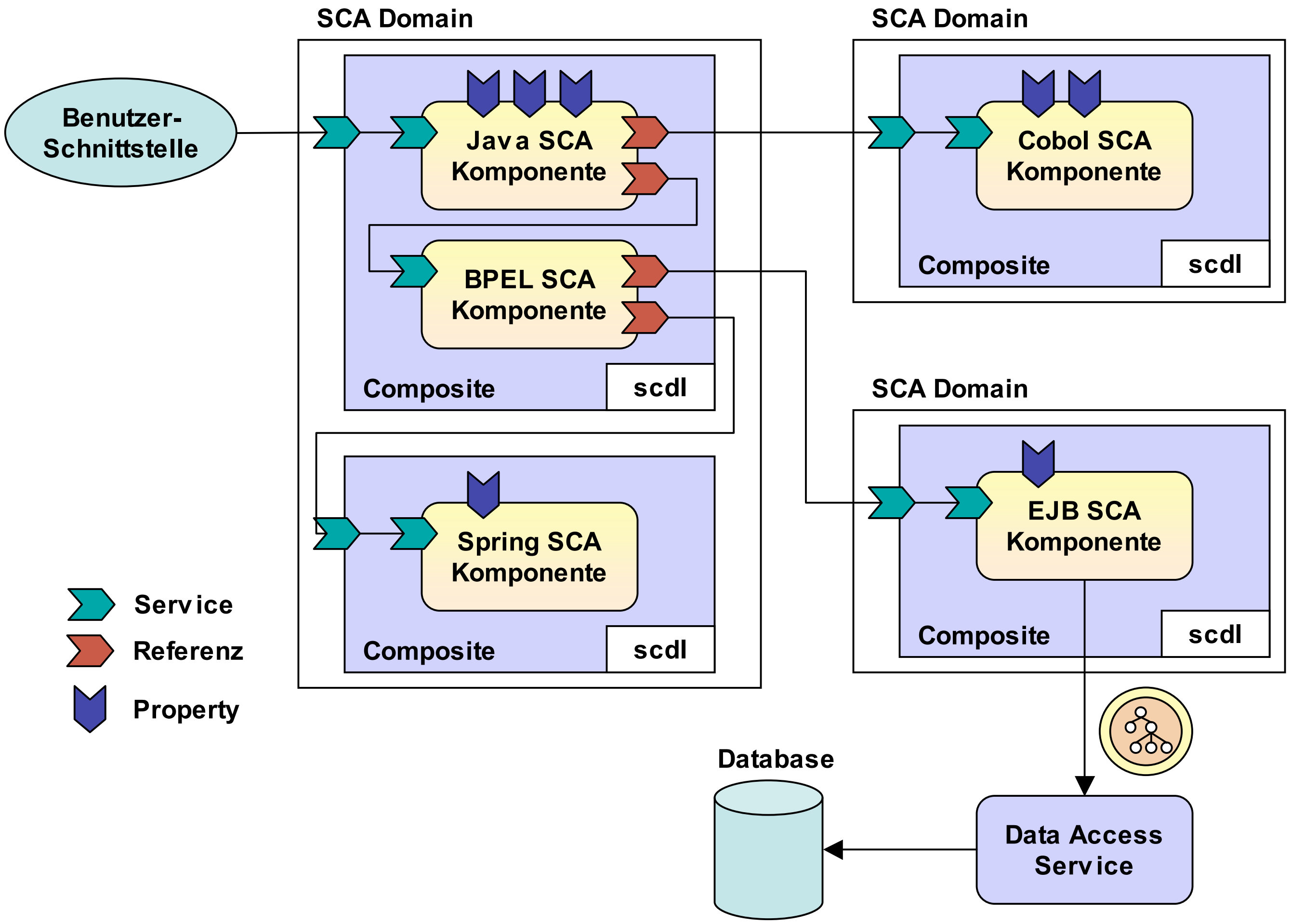
Orchestration or Choreography

- Choreography
  - Describe Interaction between Services
  - E.g. WS-CDL
- Orchestration
  - Process Oriented
  - E.g. BPEL
  - Describes Business Process and Interaction with Services (on basis of WSDL)

Service Component Architecture (SCA)

- Similarity to CORBA Component Model
- Platform-independent Standard for Service Composition using Runtime
- Series of Standards driven by *BEA, IBM, Sun, Tibco, Oracle, SAP, Siemens, Software AG, Interface21, ...*
- SCA describes
  - Service Interface
  - Properties (set at runtime)
  - References
  - Implementation of Service
- A composite runs in one runtime environment and uses **`SCDL`** for configuration
- “Heavy” use of Annotations (in Java)
- Implementation e.g. Apache Tuscany (Incubator)

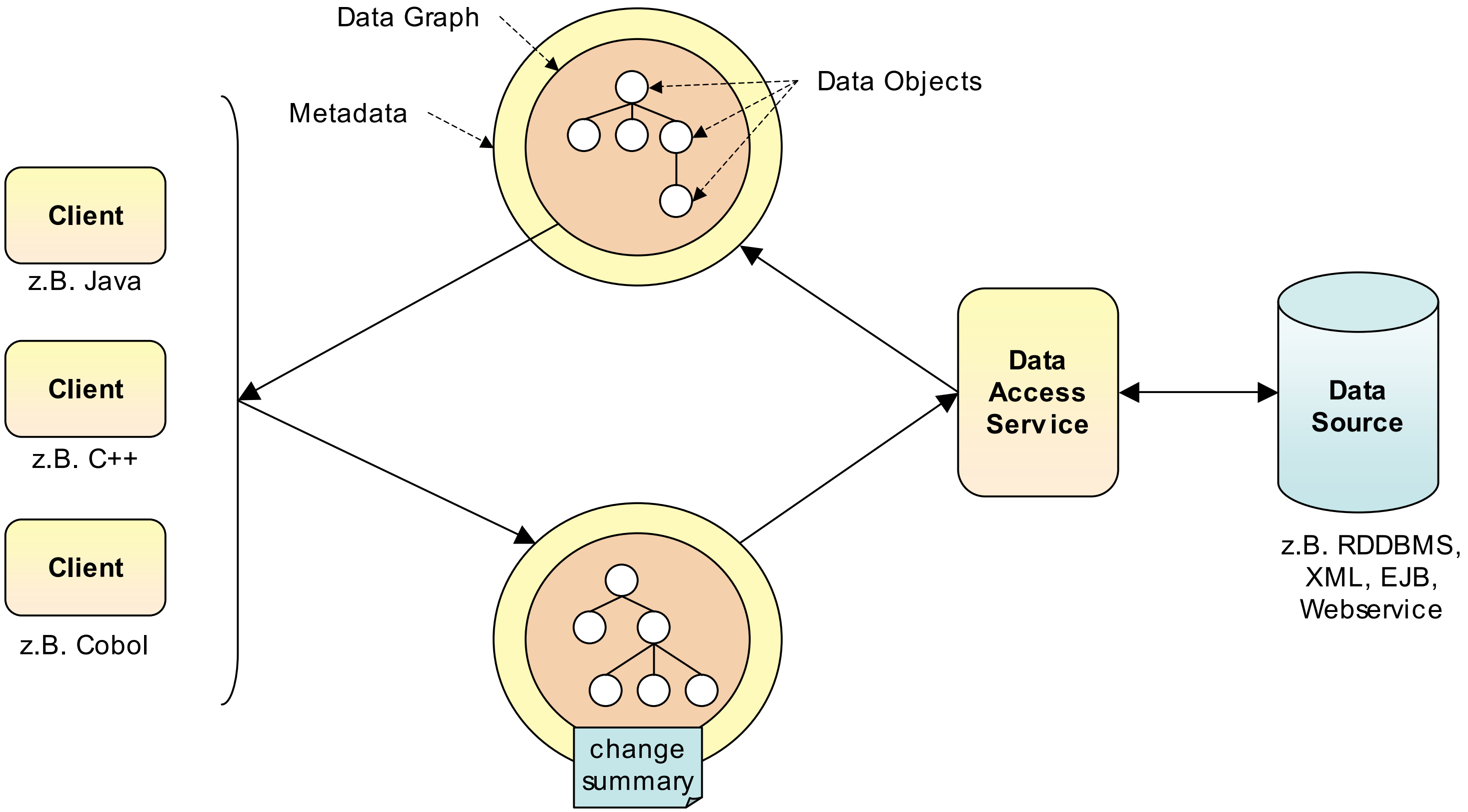
SCA Example


SCA Domain
SCA Domain
Benutzer-Schnittstelle
Java SCA Komponente
Cobol SCA Komponente
BPEL SCA Komponente
Composite
scdl
Composite
scdl
SCA Domain
EJB SCA Komponente
Spring SCA Komponente
Service
Referenz
Property
Composite
scdl
Composite
scdl
Database
Data Access Service

Service Data Objects (SDO)

- Service Data Objects are part of the SCA standardisation effort but actually can be used also in other contexts
- Platform-independent data-structure description
- Metadata for data-structure
- Statical and dynamical typing
- **Data objects** and **Data Graphs**
- Validation and Constraints
- Disconnected Model (!)
- Different Mappings supported (e.g. to XML Schema)
- Change History

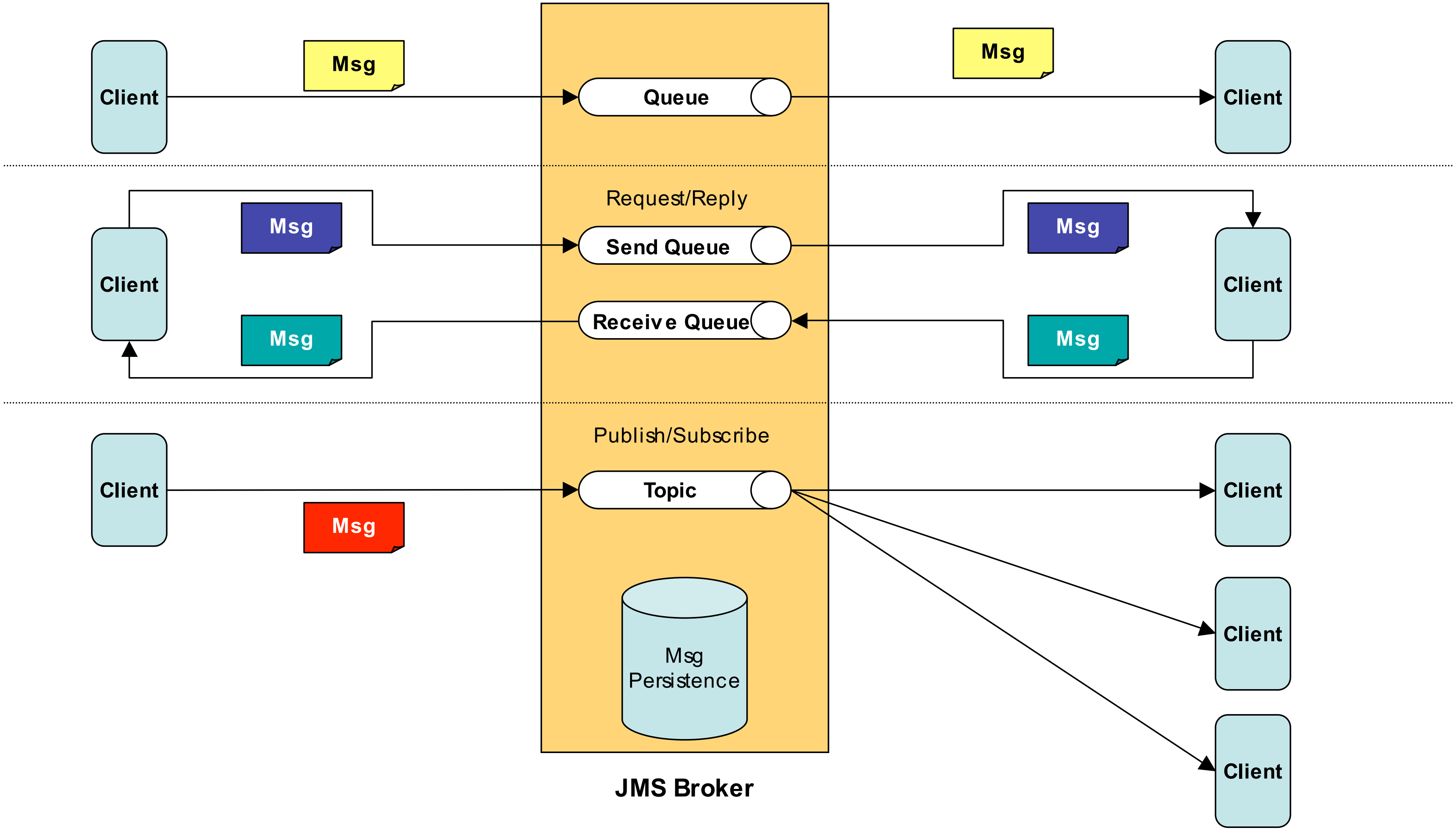
SDO Example


Data Graph
Metadata
Data Objects
Client
z.B. Java
Client
z.B. C++
Client
z.B. Cobol
Data Access Service
Data Source
z.B. RDDBMS, XML, EJB, Webservice
change summary

“Evolutionary” Approaches

Aggregation using Enterprise Integration Patterns and Messaging

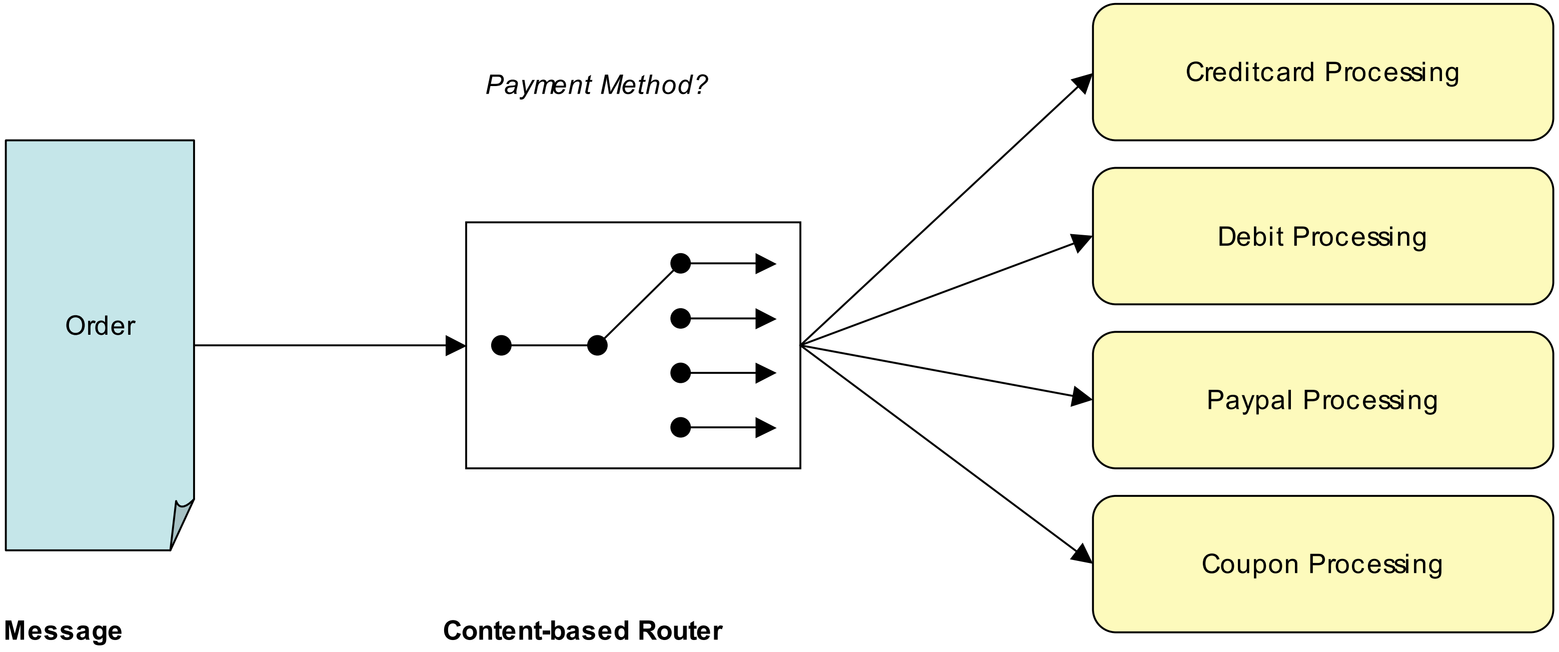
Basis: Java Messaging Service Broker



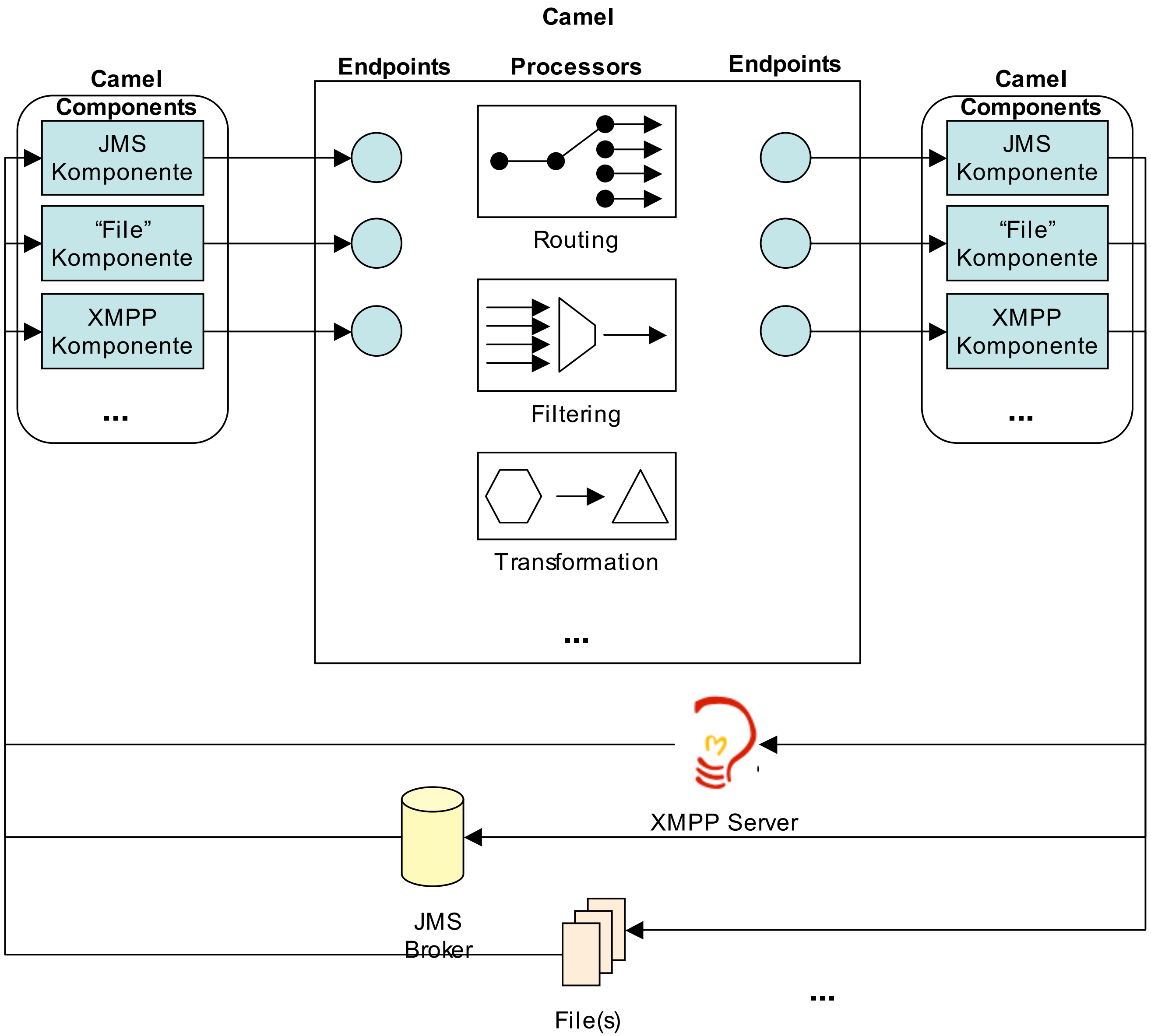
Enterprise Integration Patterns

- See Website and Book from Gregor Hohpe: http://www.integrationpatterns.com/
- Patterns to provide guidance in many “typical” enterprise integration scenarios
- Organisation of Patterns
  - Integration Style
  - Channel Patterns
  - Message Construction Patterns
  - Routing Patterns
  - Transformation Patterns
  - Endpoint Patterns
  - System Management Patterns

Example: Content-based Router



Example Setup : MOM, ESB, Apache Camel



Event Driven Architectures

Event Driven Business

- Real World Systems get more and more enriched with “sensory components” like RFID
- Every process step in the real world (particularly when multiple companies are involved) has an according step in the IT System (traditionally paper-flow)
- Towards Event-Driven Architectures:
  - Events in “real world” trigger IT Systems

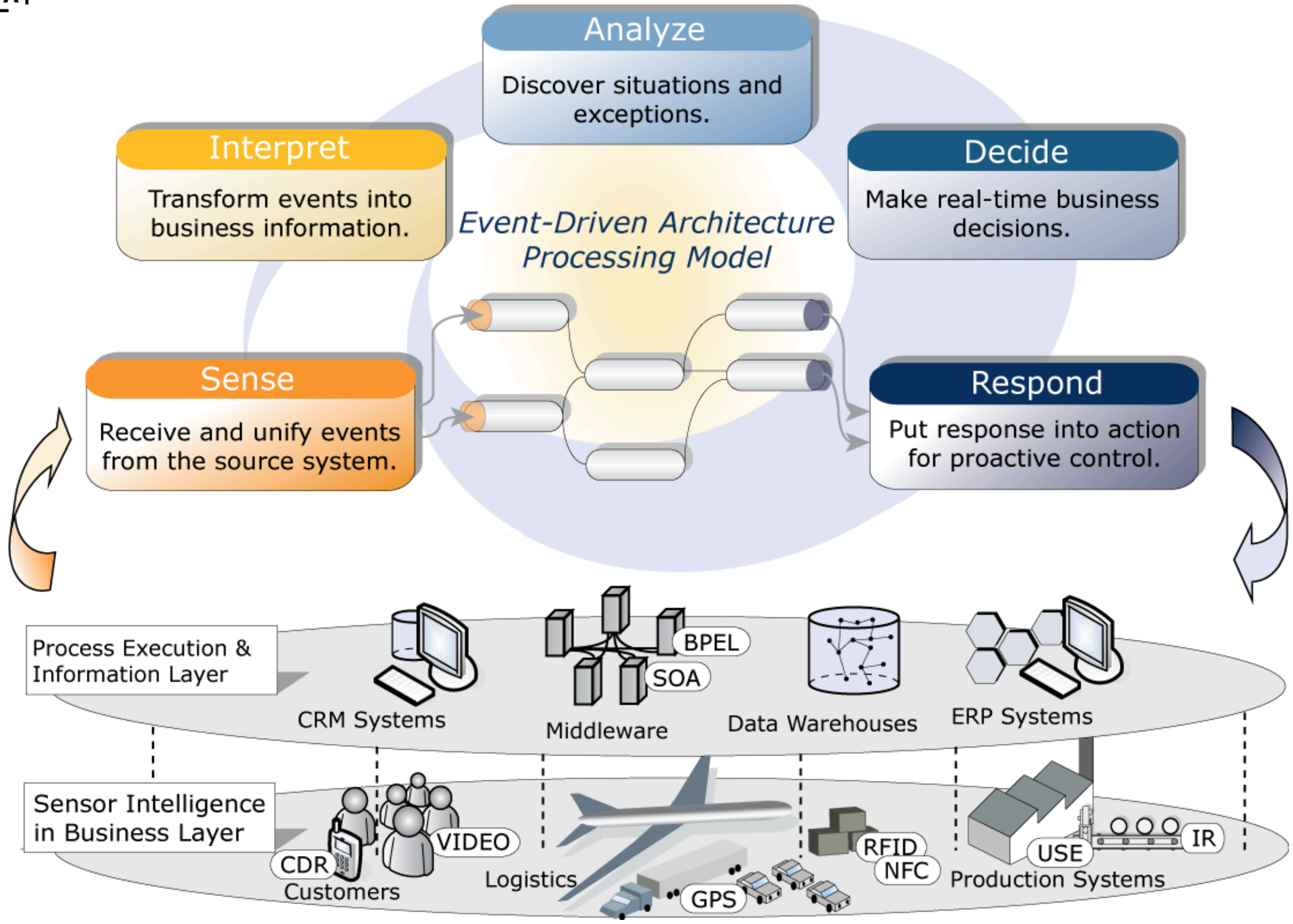

Analyze
Discover situations and exceptions.
Interpret
Transform events into business information.
Decide
Make real-time business decisions.
Event-Driven Architecture Processing Model
Sense
Receive and unify events from the source system.
Respond
Put response into action for proactive control.
Process Execution & Information Layer
CRM Systems
BPEL
SOA
Middleware
Data Warehouses
ERP Systems
Sensor Intelligence in Business Layer
CDR
VIDEO
Customers
Logistics
GPS
RFID
NFC
USE
IR
Production Systems

Figure by Josef Schiefer, Senactive

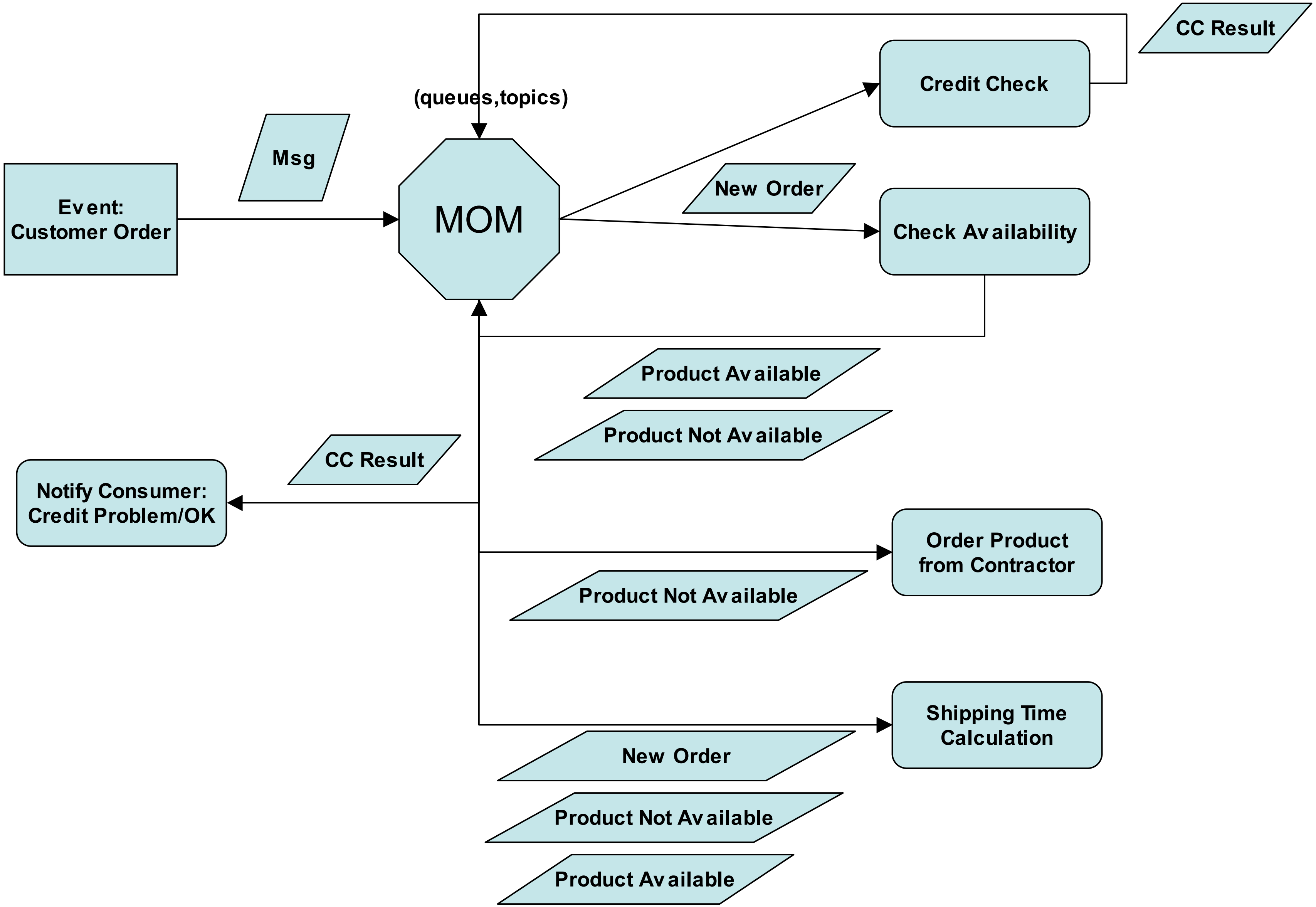
Showcase Requirements

- Webshop
- Customer orders products
- Check customer credit
- Check availability of product(s), order products from contractor(s) if required
- Calculate shipping time
- Start delivery procedure, invoke other systems like ERP, CRM, logistics systems
- Notify Customer about progress
- Perform delivery
- Charge Customer

Traditional Architecture

- Central controler component
- Interacts with several services to perform tasks
- ...
- ...
- ...

Event-Driven Architecture (motivational example-fragment)



Syndication Scenarios

Syndication?

- Integration of different information pieces
  - into unified UI environment
  - for post processing and creation of syndicated content
- Separation of Roles
  - Content producer
  - Content integrator
  - Content publisher
  - ... ?
- Standards?
  - Based on Web Standards: http, REST, Atom, Atom Publishing

Levels of Syndication, “Mashup”

- “Private”, Individual
  - Integration of News Feeds
  - Email
  - Browser Integration
  - Newsreaders
  - Usage of Portals (e.g. Yahoo, Pricefinder...)
- Enterprise “Mashup”
  - E.g. content portals (e.g. Yahoo, Google News)
  - Inclusion of content from dedicated content providers
  - Integration of information from various parts of the company for publication
  - “Meta-Enterprise-Information” (e.g., Pricefinder)

Conclusion

- Identification of Services (logical level) first step
- Decision for proper standard base!!
- Identification of Architectural Styles to be used
- Identification of non-functional requirements (security, service-level agreements, ...)
- Definition of “Middleware-Landscape”, ESB, MOM, Registries ...
- Identification of Aggregation Needs (depending on application scenarios)
  - Process based
  - “Mashup”
  - Event-driven
- Implementation of Aggregation with according standards

Thanks for the Attention!

Dr. Alexander Schatten

Institute of Software Technology and Interactive Systems,TU-Wien

http://www.schatten.info
alexander@schatten.info

http://best-practice-software-engineering.blogspot.com/