Agenda

1. **SOA**
   - Introduction
   - **Conceptual background**
   - Technologies / Platforms
2. Experiences: Real World SOA Projects
3. Selected Best Practices: Tips on SOA and Design
4. Conclusion

Distribution models

Client / Server

- Partitioning of the incurred processes in two (overlapping) groups
  - 'Server' implements a defined service
    - Waits for request
    - Sends response
  - 'Client' asks the server for a service
    - Sends request
    - Waits for response
  - Also used: 'Servant'
    - At the same time server and client
- Communication between client and server
  - Connection-less protocol (efficient, but less reliable)
  - Connection-oriented protocol (reliable but expensive)

Three typical Logical Application Layers

- User level / User interface layer
  - Interaction with human users
  - Preprocessing / Control of data (e.g. forms)
  - Running directly at the user (as a application or at the Web browser)
- Processing layer
  - Implements the application logic and the control flow
  - Provides the user interface with data from the data level
  - Running directly at the user or at the net on a server
- Data layer
  - Stores data persistently (e.g., in files)
  - Might guarantee consistency of data (e.g., via transactional DBMS)
  - Organizes data (hopefully …) independent from applications

N-tier Architectures

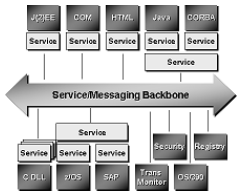
- 2-tier-architecture (client/server)
  - Client contains user interface (or a part of it)
  - Server contains processing and data level
- 3-tier-architecture
  - Client contains user interface (or a part of it)
  - Server will separated in
    - Application server
    - Database server
- Multi layer architectures (Multi-tier-architectures)
  - Client and server are separated into N layers
  - Separation depends on the application

Service-oriented architectures

Service-orientation

- Architecture principle:
  - Horizontal separation of the functionality into a
    - logically encapsulated,
    - communicating
    - 'completely' described

    Service



- Some immediate questions:
  - Granularity: How and what will logically be encapsulated?
  - Coupling: How and with what is what communicating?
  - Directness: What is described in which complexity?
  - Design: How will a SOA be implemented?
  - Technology: With what will this be implemented?

Principles

- Granularity
  - Which logic contains a service?
  - What is the adequate granularity?
  - How is the processing time?
  - How large is the required data volume per time?
- Coupling / communication
  - What must A know to communicate with B?
  - Loose coupling (as little as possible , as much as necessary)
- Directness
  - What is the content of the service description?
    - Name
    - Location
    - Data type
    - Interchange format
    - Technical communication channels

Design aspects

- Architecture principle: „Separation of Concerns“
  - Dijkstra, 1974
- Applicable with
  - Modular programming
  - Object-orientation
  - Component model
  - Service-orientation
- Basic principles of a SOA (where possible)
  - Re-Use
  - Service contract
  - Loose coupling
  - Abstraction
  - Composable
  - Autonomy
  - Be stateless
  - Detectable

Communication

Communication models

Coupling / Communication

- Communication
  - Two processes A and B exchange data
  - Communication style is
    - Corporate repository (A and B at the same machine)
    - A network (A and B at different machines)
- Network
  - Serves the communication
  - Is unreliable (particularly the internet)
  - Has latency time
- Protocol
  - Rules and standards for the communication
  - Connection-less
  - Connection-oriented
- Remote Procedure Call, RPC
  - Invoke a procedure at another machine
  - Alternative: Doors, asynchronous RPC, delayed synchronous RPC
- Remote Method Invocation, RMI
  - Invoke methods of a persistent or transient object at another machine
- Message oriented Middleware, MOM
  - Enable persistent asynchronous communication
- Streams
  - Time-critical and time-dependent continuously communication

Message oriented communication

Persistency

- *Transient communication*: A Message will be discarded by a communication system, when the receiver is not ready
- *Persistent communication*: A Message will be stored by a communication system until the receiver got the message

Synchronicity

- *Synchronous communication*: The sender blocked, till the message is delivered or even until the message is completely processed
- *Asynchronous communication*: The sender works along, right after the message is received from the communication system

Communication

Communication pattern

Communication pattern

- A communication pattern (MEP = *message exchange pattern*) defines a message sequence for the communication
- Different pattern for miscellaneous application areas
- Basic communication patterns:
  - Request/Response
  - Fire and Forget
    - Single-Destination
    - Multicast
    - Broadcast
- Complex communication pattern
  - Publish/Subscribe with / without a broker

Coordination

- Orchestration
  - Control of a workflow
  - Overcoming of incompatibilities of the participants
  - Recycling in a new context
  - Orchestration in SOA
    - Standardized interconnection of services to a workflow
    - Result is a service
    - Specialized orchestration logic for services
- Choreography
  - Collaboration process between two or more equal partners in a ambience that is not controlled by one of the partners
  - Interoperability at focus
  - Message exchange
  - Execution of complex activities by several exchanged messages
  - Modularization enables re-use

Coordination: orchestration vs. choreography

- Orchestration:
  - Defines a organization specific workflow
  - Controlled by an organization, also when external services are called
- Choreography
  - Not controlled by an organization
  - Bypass organization specific differences

Service-orientation in an Enterprise

- Extends enterprise logic
- Services provide/are an abstraction layer between
  - Business logic and
  - Application logic
- Services modularize enterprises
  - Independent logical units
  - Connectivity layer
- Layering of services
  - 'Parent-services' encapsulate 'child-services'
  - Provides
    - further abstraction
    - Re-use potential

Abstraction layers

The service layer

- Questions:
  - Which logic must be represented by a service?
  - How can services be in relationship with the application logic?
  - How can services represent business processes?
  - How can services be build and positioned to react fast to changes?

Abstraction layers

- Abstraction inside the service layer helps to answer the questions
- Three abstraction layers
  - Orchestration service layer (contains orchestration services)
  - Business service layer (contains business services)
  - Application service layer (contains application services)
- The layering is enabled by the use of the eight basic principles of a SOA

Application Service(s) Layer

- Contains application services
  - Technology-specific functionality
  - Data processing in new or legacy applications
- Normal characteristics of application services
  - Offer functionality inside a specific processing context
  - Sustain on existing resources on a platform
  - Are solution agnostic
  - Are generic and recyclable
  - …

Frequent application service sub-types

- Wrapper-services
  - Encapsulation of an existing, productive (sometimes called: 'legacy') application in parts or complete
- Hybrid services
  - Contains application and business logic (Legacy)
- Utility services
  - Solution agnostic implementation of functionality
  - Recyclable operation for orchestration
- Integration services
  - Composition on the application service layer
  - Aggregation of fine-granular applications to a raw-granular service

Business Service(s) Layer

- Contains business services
  - Implementation of business logic as services
  - Representation of the enterprise business model
- Two typical (models) for business services
  - Job centric business services
    - Orientated at a job or a process
    - Marginal potential for re-use
  - Entity centric business services
    - Entity = Unit (Structure or data set) of a business event
    - Sample: Offering, booking, bill, time sheet, account
    - High potential for re-use
    - Can be used by job centric business services

Orchestration Service(s) Layer

- Highest abstraction layer
- Contains process services
  - Control of the execution sequence of the operations
  - Orchestration of subjacent services
    - Mapping of business processes as a workflow
    - Connector between business processes and services
    - Technology-independent modeling
  - Coordination of the involved
- If a process service is re-uasble as a whole it might become a utility service
- Orchestration requires specific software (WfMS/Process Execution Engine).
  This software might not exist (yet) in an enterprise

End – SOA Conceptual Background

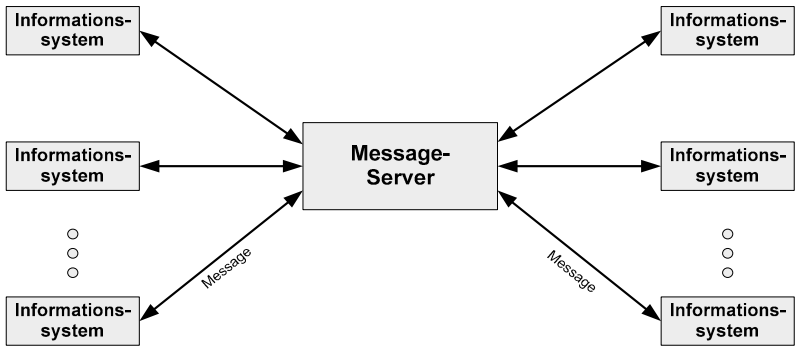
Agenda

1. **SOA**
   - Introduction
   - Concepts
   - **Technologies / Platforms**
     - **Foundation and older (but still often used) technologies**
2. Experiences: Real World SOA Projects
3. Selected Best Practices: Tips on SOA and Design
4. Conclusion

Realizing SOAs – Technologies

- SOAs are frequently associated only with Web Services & ESBs
- However, distinguish between concept and implementation technology
- SOA is an abstract concept (if not even a philosophy)
- Many implementation technologies are usable, such as:
  - Messaging & TP monitors
  - CORBA
  - Java EE
  - Microsoft .Net
  - Enterprise Service Bus (ESB) / SOA Platforms
  - Web Services
  - ….
- Let‘s look at a few of them
  - Note: Web services / ESBs / SOA Platforms are currently the dominating ones
    - Thus the tutorial describes other examples very briefly only

Foundation Technology: Message oriented middleware (MOM)



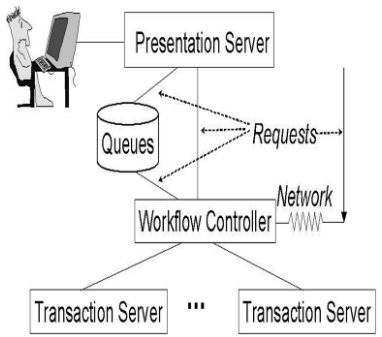
Asynchronous messages (if necessary with transactional / delivery control) within the server.

MoMs around, e.g.

Active MQ, JMS, Notification, Sonic MQ, WebSphere MQ

Foundation Technology: TP-monitors: Core Components

Bernstein, Newcomer 1997

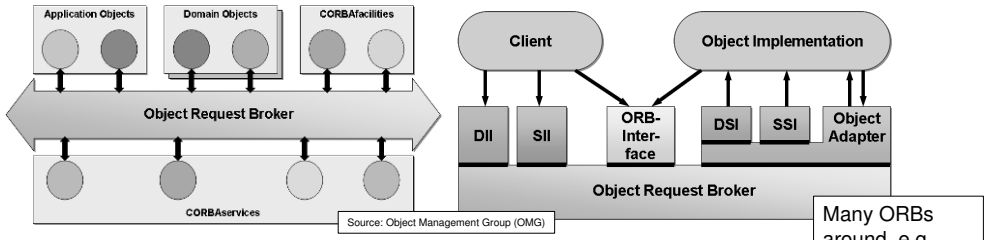


- Typical features
  - Authentication & authorization of clients
  - Sequence control by (micro) workflow controller
  - Load balancing
  - Fault tolerance, e.g., by multiple TP-monitor-processes
  - Several communication styles
  - Exception Handling
  - Safepoints
  - Directory service
  - Logging
  - Integration with system management
  - ACID-transactions, two-phase-commit

TP monitors around, e.g.

X/OpenDTP standard, CICS / IMS, Tuxedo
JTA/JTS
OTS
MTS

Object **M**anagement **A**rchitecture (OMA)
Common Object Request Broker Architecture (CORBA)
A foundation for 1st generation SOAs



Source: Object Management Group (OMG)

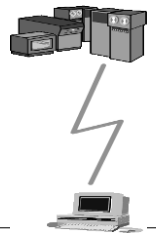
Many ORBs around, e.g.

JavaSDK Orb, Orbix, Orbacus, TAO, VisiBroker

OMG Interface Definition Language (OMG IDL)

```
interface Account
{
  readonly attribute float balance;

  void credit (in float amount);
  void debit (in float amount);
}
```

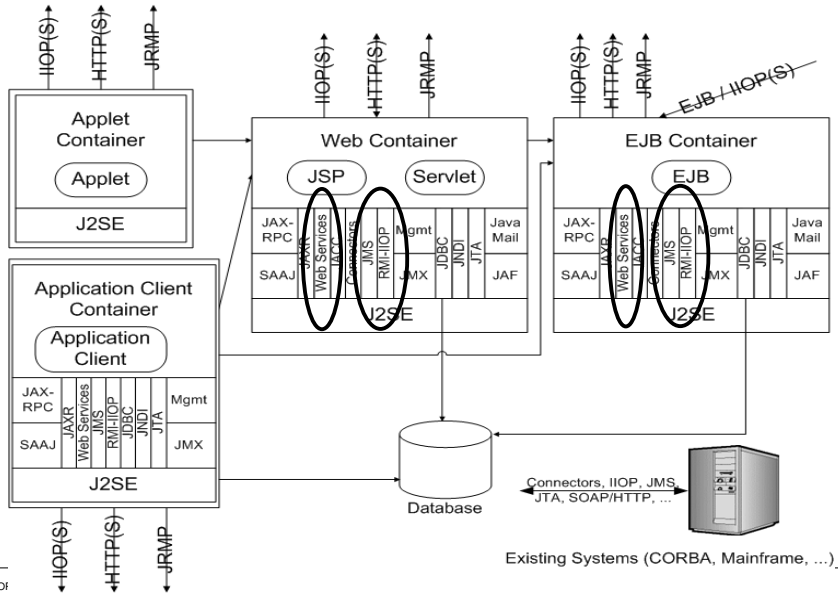
OMG Internet Inter-ORB Protocol (IIOP)



Java (2) Enterprise Edition (J2EE / Java EE) & Enterprise Java Beans (EJB)

- Idea
  - Unify proven technology concepts
    - TP-Monitors
    - Distributed objects & components
    - Application server
  - Standardized API
    - "Standard contract" between application server and components (EJBs)
  - Standardized framework
    - Portable
    - Transaction oriented
    - Reusable
    - For "the typical 60-70% of DB-style applications" much less infrastructure knowledge / development required

Java EE / J2EE-Standard
J2EE 1.4 Platform (current version similar: Java EE >= 5 / 6; even more parts)



Responsibilities: Container & Components

- Container **enables** Quality of Service
  - Concurrency
  - Consistency
  - Security
  - Availability
  - Scalability
  - Administration
  - Integration
  - Distribution
- Developers **focus on** Business components
  - Presentation
  - Logic
  - Optional: data access

Java EE Services

- Java Naming and Directory Service (JNDI)
- Security
- Java Transaction Service (JTS), Java Transaction API (JTA)
- Java Mail
- DB Connectivity (JDBC)

- Key ‘SOA’ / Integration Services in Java EE
  - Java EE Connector Architecture
  - Java Messaging Service (JMS)
  - Java IDL & Reverse mapping
    → Full CORBA integration in Java EE
  - Web Services

. . .

Agenda

1. **SOA**
   - Introduction
   - Concepts
   - **Technologies**
     - **Web Services**
     - **SOA Stack / Platform Examples / REST**
2. Experiences: Real World SOA Projects
3. Selected Best Practices: Tips on SOA and Design
4. Conclusion

Remember: Definition Web Service

„A Web service is a software system designed to support interoperable **machine-to-machine interaction over a network**. It has an **interface described in a machine-processable format** (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using **SOAP messages**, typically conveyed using **HTTP with an XML serialization** in conjunction with other Web-related standards." (W3C Web Services Architecture Group, http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/)

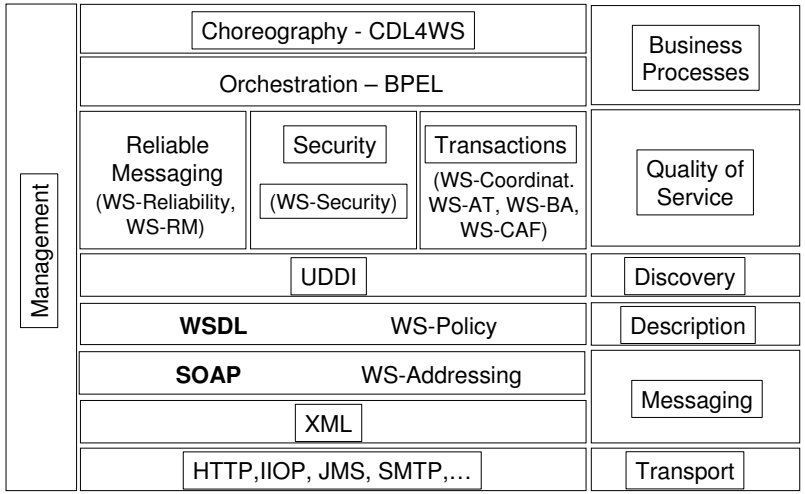
Web Services: Standardization

- Three organizations drive Web Services standardization
- The World-Wide Web Consortium (W3C, http://www.w3.org) cover core technologies in four working groups
- OASIS (Organization for the Advancement of Structured Information Systems, http://www.oasis-open.org) covers processes, user interaction with and composition of Web services
- WS-I (Web Services Interoperability Organization) defines WS interoperability with profiles and test suites

WS-* Standards ! (Too) Many and REST ?

http://www.innoq.com/resources/ws-standards-poster/

Focus here: ‘Core’ Web Services Technology Stack



Web Services Description: WSDL

- WSDL
  - Web Services (interface) Description Language
- Basic idea
  - Communication partners exchange message to ‘invoke’ a Web services and to get back ‘results’
- Abstract part
  - Data types and messages types
  - Service operations
- Concrete part
  - Concrete service endpoint (URL) including access protocol (e.g., HTTP)

Sample WSDL – Stock Trading Price Query

```
<?xml version="1.0"?>
<definitions name="StockQuote"
          targetNamespace="http://example.com/stockquote.wsdl"
          xmlns:tns="http://example.com/stockquote.wsdl"
          xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
          xmlns:xsdl="http://example.com/stockquote.xsd"
          xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
          xmlns="http://schemas.xmlsoap.org/wsdl/">

    <message name="GetTradePriceInput">
        <part name="tickerSymbol" element="xsd:string"/>
        <part name="time" element="xsd:timeInstant"/>
    </message>

    <message name="GetTradePriceOutput">
        <part name="result" type="xsd:float"/>
    </message>

    <portType name="StockQuotePortType">
        <operation name="GetTradePrice">
           <input message="tns:GetTradePriceInput"/>
           <output message="tns:GetTradePriceOutput"/>
        </operation>
    </portType>
```

Message formats

Interface definition

… part 2

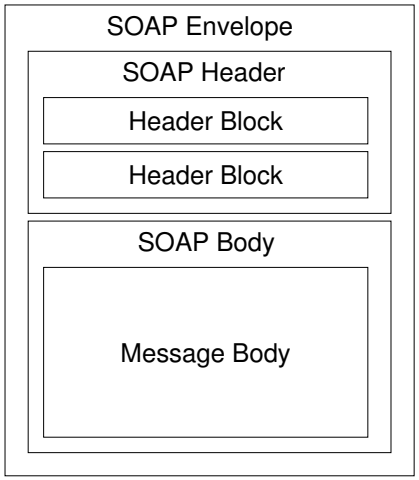
```
    <binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType">
        <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="GetTradePrice">
           <soap:operation soapAction="http://example.com/GetTradePrice"/>
           <input>
               <soap:body use="encoded" namespace="http://example.com/stockquote"
                          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
           </input>
           <output>
               <soap:body use="encoded" namespace="http://example.com/stockquote"
                          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
           </output>
        </operation>
    </binding>

    <service name="StockQuoteService">
        <documentation>My first service</documentation>
        <port name="StockQuotePort" binding="tns:StockQuoteBinding">
           <soap:address location="http://example.com/stockquote"/>
        </port>
    </service>
</definitions>
```

End point information

Concrete implementation details of operation

SOAP Messages

- A so called envelope encloses the SOAP message.
- The envelope contains an optional header and the actual message in the SOAP Body.
- Task of the header:
  - Metadata for communication
  - E.g. about transaction contexts, authentication, authorization, routing and delivery information



SOAP Method Invocation: 'Stockqoute'

```
<SOAP:body>
   <op:GetTradePrice xmlns:op="http://example.com/stockquote”>
      <tickerSymbol> GM </tickerSymbol>
      <time> 1999-05-31T13:20:00-05:00 </time>
   </op:GetTradePrice>
</SOAP:body>
```


Directory – UDDI

- Q: How to find a service ?

  ➔ Directory service(s)

- **UDDI** (Universal Description Discovery and Integration)
  - Includes standards to publish and find Web services
  - Idea: “Unifies white and yellow pages“
  - Initially founded end 2000 by Ariba, IBM, and Microsoft
  - Global UDDI Directories rarely used in practice
  - Local UDDI Directories used in enterprise integration
- Interfaces
  - Manually with browser
  - Automatic using a Web services API

WS-BPEL (Ex BPEL4WS)

- Business Process Execution Language
- Technology idea: Principles of workflow processing
  - Asynchronous messaging
  - Monitoring
  - Compensation
  - SOAP / WSDL for communication
  - XML / XPath for information processing
- A SOA business application just becomes an extensible collection of BPEL orchestrated Web services
  (+ User Interface ➔ BPEL4PEOPLE, WS Human Interaction)

BPEL Code Sample

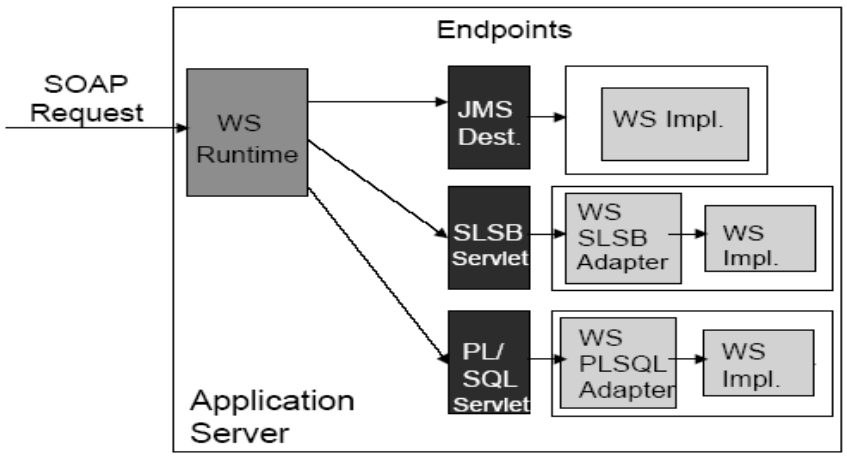
```
<flow>
  <links>
    <link name="receive-to-assess"/><link name="receive-to-approval"/>
    <link name="approval-to-reply"/><link name="assess-to-setMessage"/>
    <link name="setMessage-to-reply"/>          <link name="assess-to-approval"/>
  </links>

  <receive name="receive1" createInstance="yes">
    <source linkName="receive-to-assess"
       transitionCondition="bpws:getVariableData('request', 'amount')&lt;10000"/>
    <source linkName="receive-to-approval"
       transitionCondition="bpws:getVariableData('request', 'amount')&gt;=10000"/>
  </receive>

  <invoke name="invokeAssessor">
    <source linkName="assess-to-setMessage"
       transitionCondition="bpws:getVariableData('riskAssessment', 'risk')='low'"/>
    <source linkName="assess-to-approval"
       transitionCondition="bpws:getVariableData('riskAssessment', 'risk')!='low'"/>
    <target linkName="receive-to-assess"/>
  </invoke>
                                          . . .
  <invoke name="invokeapprover">
    <source linkName="approval-to-reply"/>
    target linkName="receive-to-approval"/>
    target linkName="assess-to-approval"/>
  </invoke>

 <reply name="reply">
    <target linkName="setMessage-to-reply"/>
    <target linkName="approval-to-reply"/>
  </reply>
</flow>
```


Agenda

1. **SOA**
   - Introduction
   - Concepts
   - **Technologies**
     - **Web Services**
     - **SOA Product / Stack / Platform Examples / REST**
2. Experiences: Real World SOA Projects
3. Selected Best Practices: Tips on SOA and Design
4. Conclusion

Web Services & Java EE/J2EE >= 1.4 Application Server

JSR 109ff, WS-I 1.x, (Some) Web Services End Points



Web Services in J(2)EE/EJB (since J2EE 1.4)

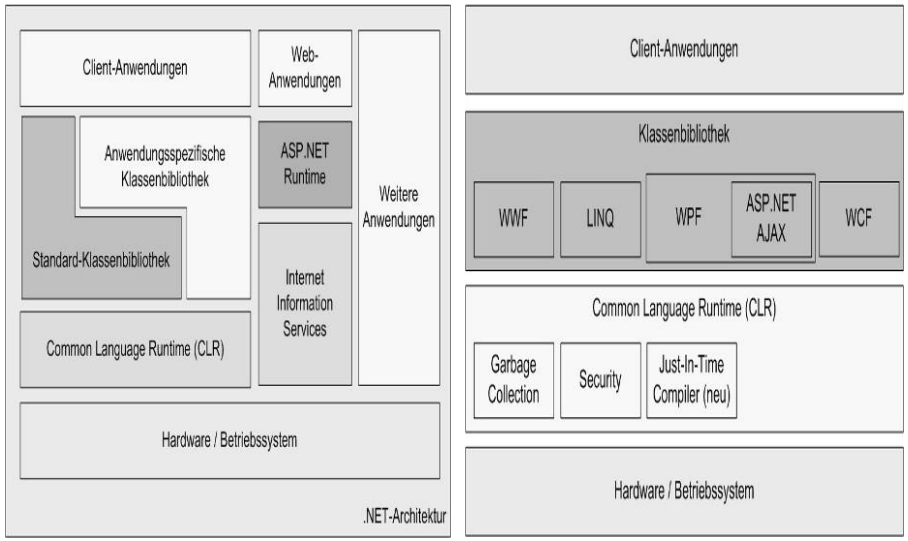
- Expose endpoints, e.g. stateless session beans (SLSB)
- Use JAX-RPC to call Web services from EJBs as clients or use Java XML binding to process XML and make direct http/SOAP calls (e.g. also using SAAJ)
- XML additions, e.g. XML-Signature & XML-Encryption exist for security purposes
- The QoS characteristics of „your“ application server are inherited (and thus typically different, e.g. regarding high availability and fault tolerance)

Web Services in EJB 3.x

- Annotations in EJBs
- @WebMethod
  … as simple annotation for Remote-capable methods in the code

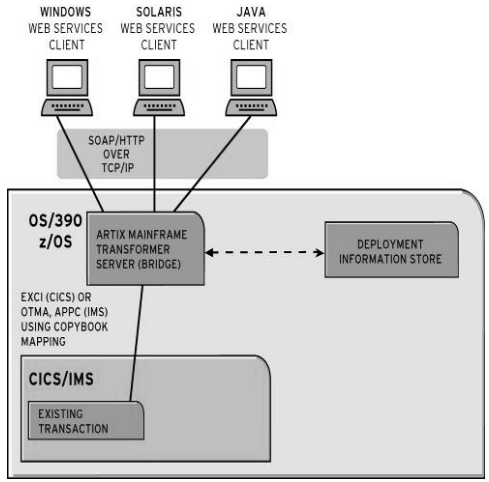
More SOA Product & Technology examples

Sample: Microsoft .NET Platform

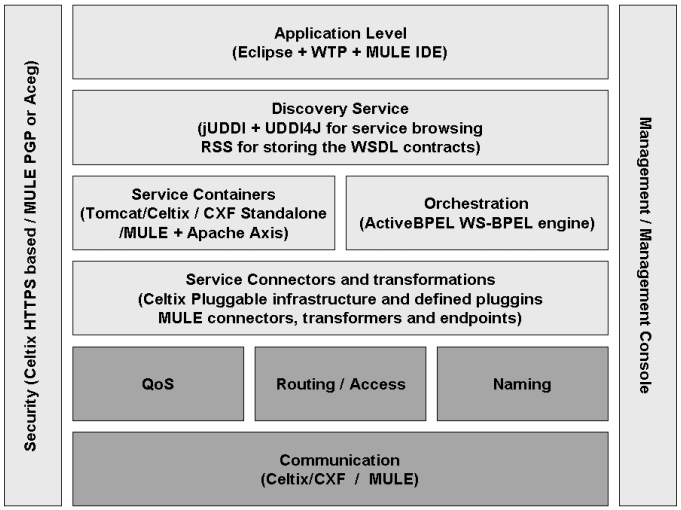


Sample: Mainframe Web Services

Source: IONA/Progress

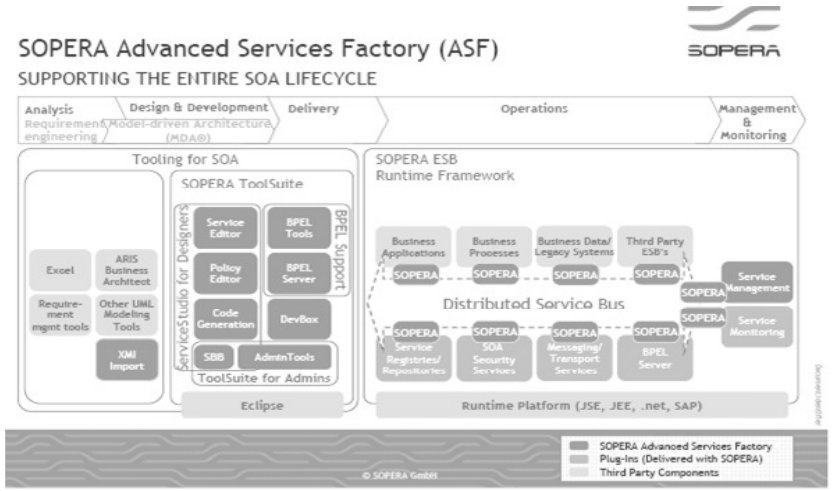


Sample: 2 'mixed' Open Source SOA Stacks



Source [SOA2007]

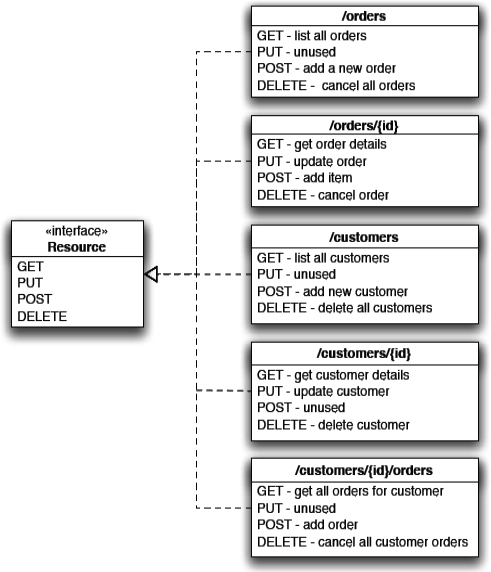
Sample: SOPERA / Talend ASF® (Origin: Deutsche Post AG)
'Core' Version: Open Source → Eclipse Project "Swordfish"



REST (1/2)

- REpresentational State Transfer
- Described by Roy Fielding in his PhD thesis
  - http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm
- One of several 'architectural styles' that uses the architectural principles of the HTTP protocol
- REST core principles
  - Identifiable resources
  - Uniform interface
  - Stateless communication
  - Resources
  - Hypermedia

REST (2/2)



- **Uniform** 'service' **interface** for all resources
- **Map**ping of generic 'HTTP **verbs**' to concrete **application semantics**
- Standardized application protocol

Fachhochschule Hannover
University of Applied Sciences and Arts

End – Implementation Technologies

Agenda

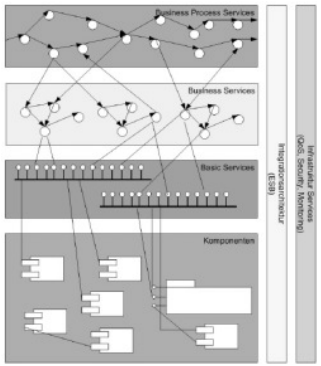
1. SOA
   - Introduction
   - Concepts
   - Technologies / Platforms
2. **Experiences: Real World SOA Projects**
3. Selected Best Practices: Tips on SOA and Design
4. Conclusion

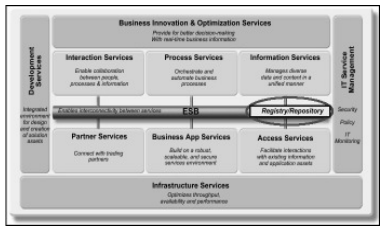
Application Samples: Web Services (SOA)

- Information Web Services (B2B, B2C)
  - Flight times
  - Stock prices
  - "Spell Check" (Google)
- "Business" Web Services (B2B, B2C)
  - Flight reservation
  - Car rental
  - Internet Auction
  - Credit cards: Authorisation / payment
- Internal application integration (EAI) between applications
  - Human resoucres
  - Bank accounts
  - Customer information
- …

SOA for medium-sized Insurance / Finance companies (CC_ITM @ FH Hannover, Germany)

SOA design guidelines **SOA repository evaluation**



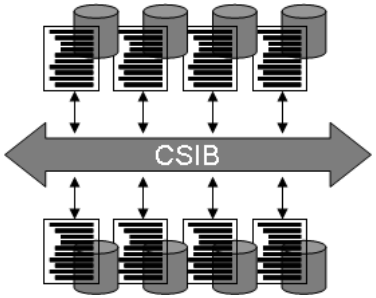


Cross Technology SOA Example Swiss-based European wide insurance

- ePlatform: Base for (mostly LAN/VPN) insurance applications
- Technically complex integration
  - Java EE: Servlets, JSP, EJB, pure Java Clients,
  - Many existing services, which need to be used, using:
    - synchronous: CORBA - heterogeneous CORBA servers C/C++, Smalltalk, Mainframe: COBOL,PL/I
    - Messaging
- Former Gartner reference architecture; OMG awards; Orbix CORBA, Java EE, Mainframe reference
- Interesting:
  Fully integrated „all-over" security incl. B1-Entry server

CS Information Bus Results – 1st Generation SOA

- **CSIB: Some Technical Parts**
  - Web Frontend
  - Java EE Application Server
  - CORBA for Mainframe Integration (mostly IMS, PL/1)



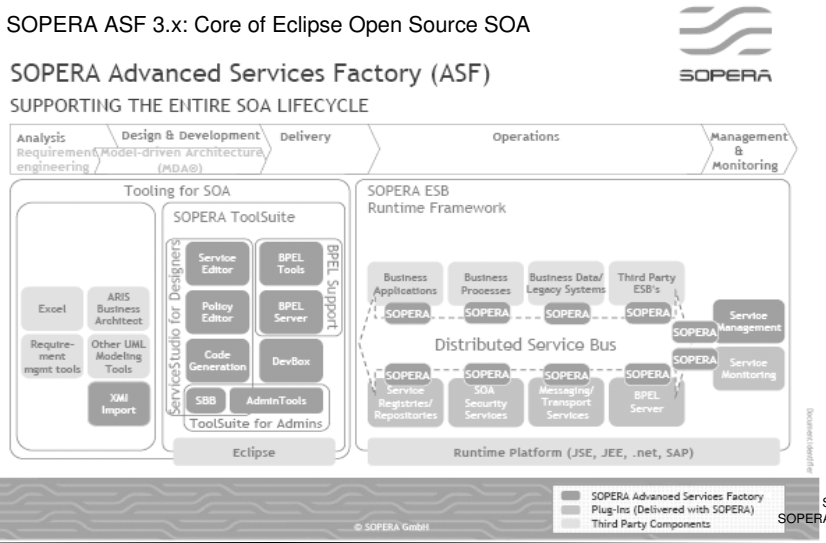
Source e.g. [Krafzig+04]

SOA-based integration – Steps

- 1999 – Analysis
  - High redundancies
  - Stovepiped solution
  - Missed business opportunities
  - High maintenance costs
- SOA based solution
  - Business separation into ‚Domains' and ‚Services' with service operations
  - Development of their own SOA platform – SOPERA (formerly Service Backbone), which became the runtime part (Swordfish) of the Eclipse SOA initiative

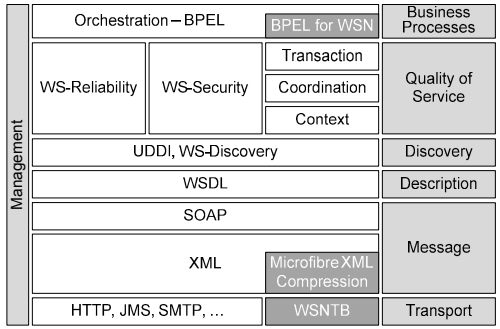
SOA-based integration – Technical viewpoint

SOPERA ASF 3.x: Core of Eclipse Open Source SOA

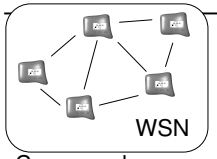


Source SOPERA 2010

Research: SOA for Wireless Sensor Networks Web Services Technology Stack

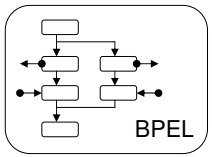


Service-oriented Operating System for Wireless Sensor Networks (WSNs)

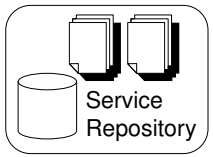


Composed WSN Application

- Goal
  - Easier application development
- Idea
  - Everything is a service
  - OS support basic services only
  - Services *migrate* at runtime
- Advantages
  - Single, simpler services
  - Easy adaptation/configuration at runtime
  - Graphical development potential
  - Enables 'self-*' (self organization)
  - Seamlessly integrates the WSN into the normal IT environment

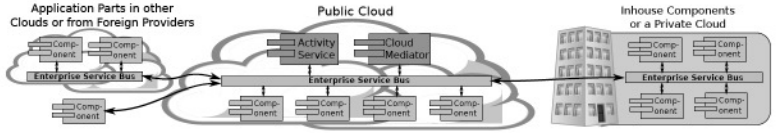


Graphical Process modelling



Service implementations

SOA-based Activity Service for Cloud Computing



Idea: Activity Service ('Trigger like')
with Active DBMS-style ECA rule semantics as service for the cloud

Basic architectural idea

- The cloud is seen as a huge SOA
- Cloud participants are services connected via ESBs / SOAP Web services
- The Activity Service is placed as a component within and for the Cloud

Fachhochschule Hannover
University of Applied Sciences and Arts

End – SOA 'Stories'

Agenda

1. SOA
2. Experiences: Real World SOA Projects
3. **Selected Best Practices: Tips for SOA Analysis and Design**
4. Conclusion

3. Selected Best Practices

SOA Design – Guidelines

- Business / Management issues
  - drive SOA through business rather than IT
    - i.e. you can't buy "SOA in an box"
  - do not throw away your running system
    - transform to SOA in an evolutionary process
- Service Definition
  - focus on long-term stable elements
    - thus develop a domain model (i.e. independent from applications)
  - choose the right granularity
    - hierarchical and coarse-grained instead of fine-grained services
  - encapsulate underlying applications & middleware

3. Selected Best Practices – See: FH Hannover / CC_ITM sample

Service Design – "Classify Services"

- Classification of services simplifies
  - common language – esp. levels of abstraction
  - effort estimation (design, implementation, operation)
- Service Type Hierarchy

Source: [Krafzig+04]

  - Basic Services
    - Data, Logic (based on components, often given)
  - Intermediary Services
    - Stateless: mediation, translators, adapters, …
  - Process-Centric Services
    - Stateful services – encapsulate business processes
  - Public Enterprise Services
    - Public services – like process-centric w. further restrictions

3. Selected Best Practices

SOA Design – Guidelines

- Clear Service *Interface* and *Semantics*
  - Design by contract preferred
  - Services must support versioning – interfaces *will* change
- Service Implementation Issues
  - Design services as open and shared
  - Distinguish synchronous vs. asynchronous services
    - think asynchronous message-oriented, document-centric
  - Do not ignore performance

Source: IONA

Service Management

- *Services are 99% Orga, 1% Technology* ("K. Cvetkovich")
- Create and Establish a *flexible and fast*

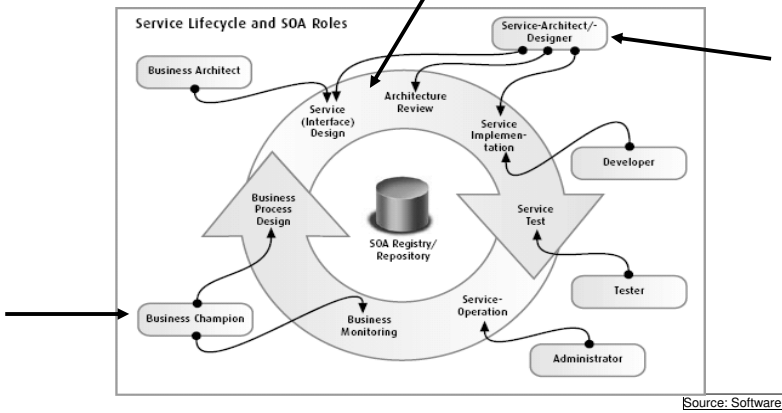
  „SOA Architecture Group,
  Service Management role
  and SOA Governance group"

  Tasks may include:

  - Technologies to implement 'your' SOA — samples done
  - SOA Analysis + Design Steps/Guidelines — → next slides
  - SOA / Service life cycle management! — → next slides
  - Several more — → not here
    - e.g. Quality of Services (QoS),
    - legal aspects,
    - . . .

Service Life Cycle Management

- SOA development follows mostly standard development practices with some additions



Source: Software AG

Goals and *basic* process

- Q: How can the requirements of the automated business processes realized by services?
- Goals / Questions:
  - Which services must be implemented?
  - What logic must be encapsulated by which service?
- Process:
  - Identify initial candidates for operations
  - Group the candidates in logical contexts (→ services)
  - Define services (avoid overlapping)
  - Identify re-usable encapsulated logic
  - Check re-usability in other contexts
  - Define well-known (initial) service compositions

Basic guidelines for service modeling

- Task centric business services:
  - Think about re-usability within a process
  - Keep cross process re-usability in mind as well
  - Check process-related dependencies
  - May be initially emulate process services without an orchestration layer
- All services:
  - Solution independent modeling of application services
  - Think about further segmentation of process steps
  - Identify process steps within clear borders
  - Beware of 'cross border infiltration'
  - Have a balanced system as a goal

Goals and *basic* process

- Service-oriented design derives physical services and their composition from the service candidates
- Goals / Questions:
  - How can this derivation be executed?
  - Which SOA-characteristic must be supported?
  - Which technology shall be used to implement everything?
- Process:
  - Choose technology
  - Define architectural borders / cornerstones
  - Identify your design standards
  - Define service interfaces
  - Identify potential service compositions
  - Base it on SOA principles

Basic design guidelines

- General design guidelines
  - Define naming conventions and execute them consistent
    - For data types, messages, operations, services and processes
  - Choose adequate granularity for the interface
    - Granularity significantly influences performance and re-usability
  - Define operations and data types extensible
    - Changes are often required after launch
    - Don‘t change but upgrade existent interfaces (remember: a versioning mechanism is a ‘must have’)
  - Identify well-known potential service users
- Define technology specific guidelines
  - E.g., ‘modular design for IDL- or WSDL-documents’

SOA: Getting Started

- Develop Realistic Sample Services
  - to prove that and how SOA successfully works.
  - improves confidence and credibility, thus simplifies discussion
  - ensures ability to deliver technical and organizational aspects
- Candidate “getting started” Projects
  - small is beautiful (but not too small)
  - simple and practically relevant services (a.k.a. „processes“), thus
    - not just technical scenarios, include business aspects
    - start with projects which prove business value
  - high probability of success (non-critical projects)
  - technical services at the end
    (however, check some QoS and “SOA walkthrough prototype”)

But: don’t forget the strategic vision – “Start small but think **BIG**”

Agenda

1. SOA
2. Experiences: Real World SOA Projects
3. Selected Best Practices: Tips on SOA and Design
4. **Conclusion**

SOA 'Importance' – Google hits # Nov. 2010

***SOA**: ca. 19.8 mio; Service Oriented Architecture: ca. 10.2 mio*

4. Conclusion

- SOA is about *business*
- Technology (mainly) works
  - Standards help (but are constantly evolving)
- *Successful* SOA *requires* strong management *support*
- *SOA*:
  - is still a topic in some flux with open questions
  - is clearly no magic – "One can't buy SOA in a box"
  - certainly comes not for free

Some references

- [Bein+08] Beinhauer, Herr, Schmidt (Herausgeber) : SOA für agile Unternehmen: Serviceorientierte Architekturen verstehen, einführen und nutzen. Symposion, 2008.
- [Dunk+08] Dunkel, Eberhart, Fischer, Kleiner, Koschel: Systemarchitekturen für Verteilte Anwendungen - Client-Server, Multi-Tier, SOA, Event Driven Architectures, P2P, Grid, Web 2.0, Hanser Verlag, 2008.
- [CC-ITM-SOA06]: CC-ITM SOA-Team, FH Hannover: SOA für Finanzdienstleister - Guidelines, 2006.
- [Conrad+05] S. Conrad, W. Hasselbring, A. Koschel, R. Trisch: Enterprise Application Integration – Konzepte, …, Elsevier/Spektrum, 2005
- [Erl04] Thomas Erl: Service-Oriented Architecture – A Field Guide, Prentice-Hall, >= 2004.
  - Erl: Service-Oriented Architecture – Concepts and Technology
  - Erl: SOA Design patterns
- [Jos07] SOA in der Praxis; Josuttis, dpunkt, 2007.
- [Krafzig+04] Dirk Krafzig, Karl Banke & Dirk Slama: Enterprise SOA: Service-Oriented Architecture Best Practices. Prentice Hall, 2004.
- [IBM_SOA_WS] IBM: Patterns: Service-Oriented Architecture and Web Services, IBM Redbook sg246303.
- [Natis03] Yefim Natis: Service-Oriented Architecture Scenario. Gartner Research, www.gartner.com/DisplayDocument?doc_cd=114358
- [SOA2007]: Starke, Tilkov (Hrsg.) SOA Expertenwissen, dpunkt, 2007, http://www.soa-expertenwissen.de
- [Sprott+04] David Sprott, Law Wilkes: Understanding Service-Oriented Architecture. Online: cbdiforum.com
- [Woods04] Dan Woods: Enterprise Services Architecture, Galileo Press, 2004. (Die SAP-Perspektive zu SOA)
- [Zi+03] Perspectives on Web Services; Zimmermann, Tomlinson, Peuser; Springer, 2003.

Some online References

- www.infoq.com: Agility / Middleware / SOA Portal
- www.webservices.org: WS / SOA portal
- www.service-architecture.com WS / SOA portal

- Standards / Blueprints
  - w3.org; ws-i.org; oasis-open.org

- Many vendors for SOA/WebServices, e.g.:
  - All Java 2 EE 1.4 / Java EE >= 5 vendors (see e.g., Java EE @ Sun/Oracle; theserverside.com)
  - IBM: www.ibm.com/developerworks/soa
  - Microsoft: msdn.microsoft.com/webservices
  - Progress: www.progress.com
  - Oracle: http://www.oracle.com/technetwork/middleware/soasuite/overview/index.html
  - Software AG: http://www.softwareag.com/corporate/solutions/soa_enablement/overview/default.asp
  - SOPERA http://www.sopera.de/en/home (formerly: Dt. Post SOP Group / SOPSOLUTIONS) → Eclipse SOA framework → www.eclipse.org/soa/ (Swordfish etc.)
  - . . .

Fachhochschule Hannover
University of Applied Sciences and Arts

That's it for now – Questions?

Arne Koschel
FH Hannover, Fakultät IV, Abt. Informatik
www.fakultaet4.fh-hannover.de
http://www.koschel-edv.de/arne_koschel_publications.htm

Fachhochschule Hannover
University of Applied Sciences and Arts

Tutorial: Service Oriented Architecture (SOA) in Practice

Prof. Dr. Arne Koschel
Univ. of Applied Sciences and Arts Hannover,
Dept. for Applied Computer Science
www.fakultaet4.fh-hannover.de
www.koschel-edv.de/arne_koschel_publications.htm

Agenda

Fachhochschule Hannover
University of Applied Sciences and Arts

1. **SOA**
   - **Introduction** / **Predictions** / **Statements** / **Facts**
   - Concepts
   - Technologies / Platforms
2. Experiences: Real World SOA Projects
3. Selected Best Practices: Tips on SOA and Design
4. Conclusion

1. SOA – Introduction, Predictions, Statements, Facts
What does SOA mean?

Fachhochschule Hannover
University of Applied Sciences and Arts

- *„A **design model** based on the concept of **encapsulating application functionality within services** that interact via a common communications protocol.“* (www.serviceoriented.ws)
- *„… a software architecture that is based on the **key concepts** of an application front end, **service, service repository**, and **service bus**.“* [Krafzig+04]
- *“… is a **paradigm** for **organizing** and **utilizing distributed capabilities** that may be under the control of different ownership domains.…* [Oasis06]
- . . .

➔ **SOA in itself is “still somewhat in flux”**

- Let‘s see, how the Oasis SOA reference model works out

1. SOA – Introduction, Predictions, Statements, Facts
What does SOA mean?

Fachhochschule Hannover
University of Applied Sciences and Arts

- Service Oriented Architecture is a distributed systems architecture
- SOA represents abstract architectural concepts for building software systems as
  - loosely coupled components
  - providing services encapsulating application functionality
  - they are described in an uniform way and
  - can be discovered and composed

What does SOA mean?

- SOA is "**business** / **business process centric**"
  - SOA decouples business applications from technical infrastructure
  - Goal: Technology has no implication on high-level application landscape!

SOA is an Architectural Style

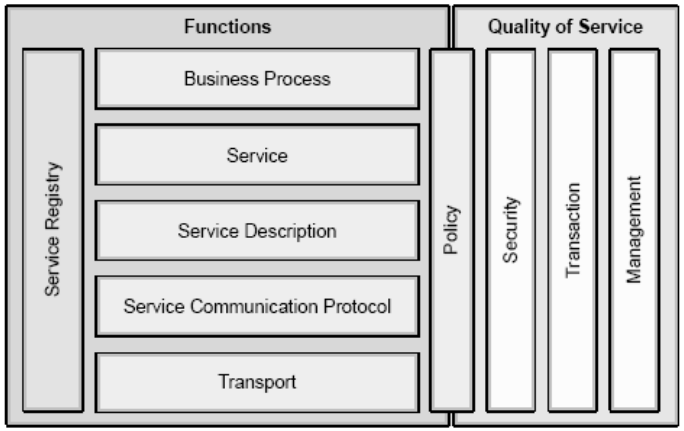
- *Service-Orientation* is an organizational principle
- based on *functional decomposition*
  - the overall functionality is implemented by services
  - no strict layering (Service implementations can use other services)
  - no centralized control entity
  - services are designed without concrete usage in mind
- Examples
  - Common "horizontal" services
    - Logging, authentication/single-sign-on, systems management, Directory lookup of services, event notification
  - "Vertical" services
    - business domain specific
    - Product feature search service, Address management, Order Status Tracking Service, Truck/trailer tracking service

SOA's potential advantages

- Potential* Advantages:
  - Improved agility of business processes
    - service orchestration
    - loosely coupling of services, data and business processes
  - Re-use of
    - existing applications or their parts
    - business processes or their parts
  - architectural benefits
    - good testability
    - support of different client-platforms
    - location transparency of services
    - separation of responsibilities during conception, development and operation

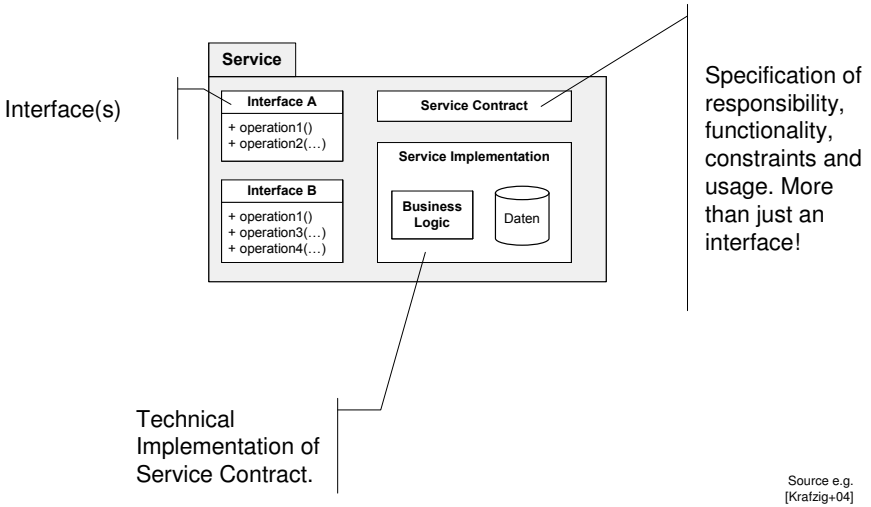
* potential, not guaranteed!

SOA – Architectural Overview



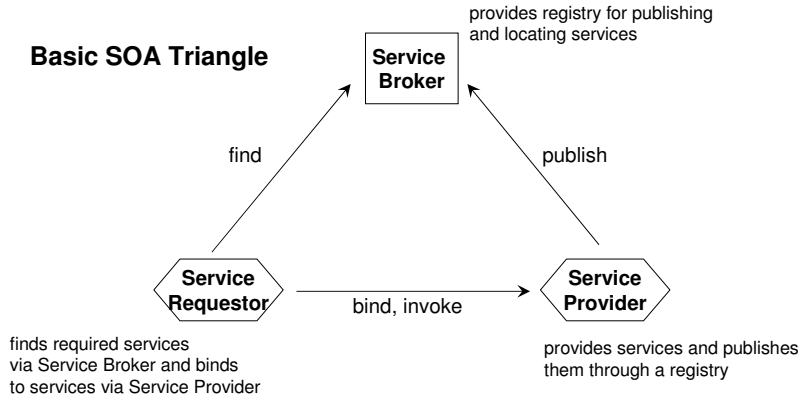
Source: [IBM_SOA_WS]

Service Specification and Implementation



Source e.g. [Krafzig+04]

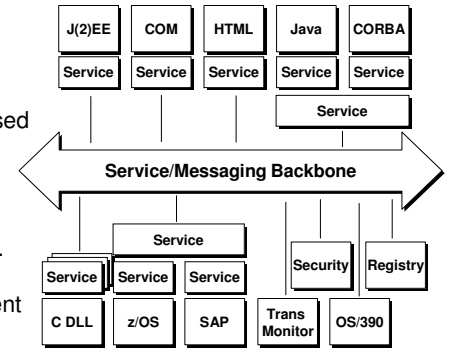
Basic SOA/Services inside: Publish-find-bind



- **but: often in "given" SOAs**
  - all services are well-known
  - therefore services are not dynamically searched and bound

SOA inside: Implementation

- Service / Messaging Backbone
  - Often: Logical / Technical "Enterprise Service Bus (ESB)"
- Services with formally described interface
- Integration of components often based on productive systems
- Different communication paradigms (Synchronous / asynchronous)
  - Request-Response; Oneway; …
- Implementation possible with different technologies / platforms, e.g.
  - CORBA, RMI, Web Services; Java EE, .Net, . . .
  - Now: Often Web Services used combined with e.g. Java EE



Sample: Web Services as SOA technology

- Definition: "A Web service is a software system designed to **support interoperable machine-to-machine interaction** over a network. It has an **interface described in a machine-processable format** (specifically WSDL). Other systems **interact** with the Web service in a manner prescribed by its description **using SOAP-messages**, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards."
  (W3C Web Services Architecture Group, http://www.w3.org/TR/ws-gloss/#WSA)
- Basic Ideas
  - Communication only via XML (SOAP)
  - Transport over established standard protocols (HTTP, ...)

Fachhochschule Hannover
University of Applied Sciences and Arts

So what – is this new at all?

Is SOA new or just "the next silver bullet"?

Fachhochschule Hannover
University of Applied Sciences and Arts

- The SOA principles aren't really new;
  e.g. remember components & connectors [Garlan/Shaw, begin 90's]
  Object Management Architecture / CORBA's [OMG, begin/mid 90's]
  ...
- However, with the Web services hype it currently gets
  "massive vendor / 'user' support"

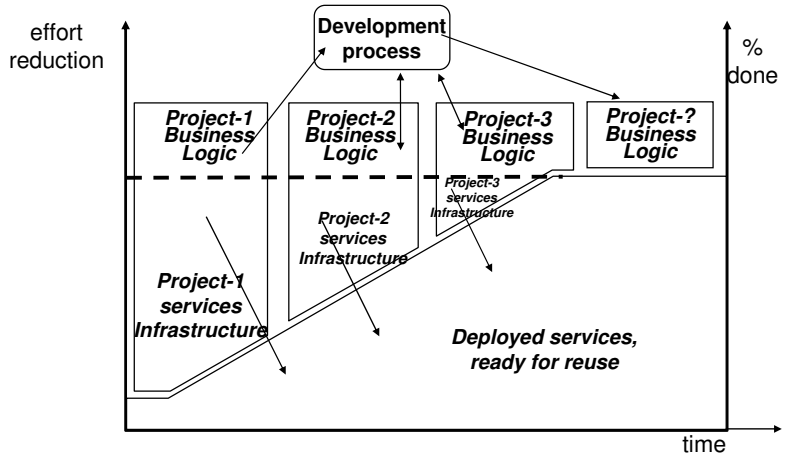
SOA Predictions for the Industry

Fachhochschule Hannover
University of Applied Sciences and Arts

- According to Gartner [Jos06]
  - SOA will become the dominant framework for creating and delivering software
  - By 2008 SOA will provide the basis for 80% of development projects
  - By 2010 80% of application software revenue growth (licenses / subscriptions), will come f. SOA-based products
  - Hype Cycle:
    - SOA is already went through the disillusion part and is now on it's way to the plateau of productivity
- Silicon.com beg. 2007: SOA is good for consultants

SOA: Potential long term value for organizations

Fachhochschule Hannover
University of Applied Sciences and Arts



Source: IONA

1. SOA – Introduction, Predictions, Statements, Facts
SOA Statements / Facts

Fachhochschule Hannover
University of Applied Sciences and Arts

- SOA is "in" for many consulting companies
  (IT and business)
- SOA/ Service Computing has it's own tracks in many IT conferences
  (industry and research)
- Many ESB / SOA / Web Services platform vendors "are out there"
  - CORBA vendors
  - Enterprise Service Bus (ESB) / Web services vendors
  - EAI vendors
  - Java EE (>= 1.4) Application server vendors
  - .NET vendor
  - Open Source projects for all of them
  - . . .

1. SOA – Introduction, Predictions, Statements, Facts
More SOA Statements / Facts

Fachhochschule Hannover
University of Applied Sciences and Arts

- SOA usage grows (some samples from Europe)
  - Type A companies have done / started SOA
    - BMW
    - Credit Suisse
    - Deutsche Post
    - Deutsche Telekom
    - ING DiBA
    - . . .
  - Some! Type B companies begin to follow
    - Medium insurance / finance companies start / explore / use
      - CC_ITM: FH Hannover & local insurance & finance companies [CC-ITM-SOA06, SOA2007]
    - Automotive partial suppliers
    - Governmental agencies

  . . .
  - Type C companies will still take some more time
    (and might only ever utilize some SOA bits)

➔ **SOA usage "still on the way"** – but it shows a clear trend

1. SOA – Introduction, Predictions, Statements, Facts
But there are other SOA Statements!

Fachhochschule Hannover
University of Applied Sciences and Arts

- SOA means
  - Same Old Architecture (remember DCE, OMG/OMA, …)
  - Stupidly Overhyped Acronym
  - Save Our Architects
- SOA is often done wrong (many praise it, some do it, few do it right)
- SOA is no silver bullet
- SOA is dead
  (but: Cloud Computing might be its strong revival ☺)

Fachhochschule Hannover
University of Applied Sciences and Arts

End – SOA Introduction